Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Національна академія наук України
Інститут проблем машинобудування ім. А. М. Підгорного

Кваліфікаційна наукова
праця на правах рукопису

ЛИТВИНЕНКО ОЛЕКСАНДР СЕРГІЙОВИЧ

УДК 519.85

ДИСЕРТАЦІЯ

МЕТОДИ ГЕНЕРАЦІЇ КОМБІНАТОРНИХ КОНФІГУРАЦІЙ ТА ЇХ ЗАСТОСУВАННЯ В МАТЕМАТИЧНОМУ І КОМП'ЮТЕРНОМУ МОДЕЛЮВАННІ ЗАДАЧ ПЕРЕВЕЗЕННЯ ТА ОБРОБКИ ВАНТАЖІВ

01.05.02 – математичне моделювання та обчислювальні методи
технічні науки
Подається на здобуття наукового ступеня кандидата наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ О.С. Литвиненко

Науковий керівник
доктор технічних наук, професор
Гребеннік Ігор Валерійович

Харків – 2018

АНОТАЦІЯ

Литвиненко О.С. Методи генерації комбінаторних конфігурацій та їх застосування в математичному і комп'ютерному моделюванні задач перевезення та обробки вантажів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук (доктора філософії) за спеціальністю 01.05.02 «Математичне моделювання та обчислювальні методи».

Місце виконання роботи: Харківський національний університет радіоелектроніки, Міністерство освіти і науки України.

Захист відбудеться у спеціалізованій вченій раді Д 64.180.01 при Інституті проблем машинобудування ім. А. М. Підгорного Національної академії наук України, Харків, 2019.

У дисертаційній роботі розглянуто методи генерації класичних та некласичних комбінаторних конфігурацій, а також застосування цих методів в математичному і комп'ютерному моделюванні задач перевезення та обробки вантажів. Об’єктом дослідження в роботі виступають процеси генерації комбінаторних конфігурацій та математичного і комп`ютерного моделювання задач перевезення та обробки вантажів, які мають комбінаторну структуру, предметом – методи генерації комбінаторних конфігурацій, математичне та комп’ютерне моделювання під час розв`язання задач перевезення та обробки вантажів.

Методи дослідження: методи комбінаторного аналізу, комбінаторної генерації та оптимізації, методи геометричного проектування для побудови математичної моделі задачі вивозу і доставки (Pickup and Delivery Problem, PDP), зокрема, метод Ф-функцій для опису тривимірних обмежень завантаження, евристичні методи побудови та аналізу рекурсивних дерев при генерації та комбінаторній оптимізації.

У роботі отримали подальший розвиток стратегії та методи генерації комбінаторних конфігурацій у частині використання рекурсивних процедур та узагальнення класів комбінаторних конфігурацій, які генеруються. Для цього було запропоновано стратегію та узагальнений метод комбінаторної генерації, на використанні яких базуються інші результати роботи. В основу методу покладено використання рекурсивної процедури backtracking для утворення одного або декількох елементів комбінаторної множини довжини i+1 від часткового елементу довжини i. В процесі своєї роботи метод будує рекурсивне дерево (backtracking tree). Універсальність запропонованого методу досягається за рахунок можливості використання різних формальних правил побудови конкретної комбінаторної множини. Запропоновано рекурсивний алгоритм *GenBase*, що реалізує узагальнений метод генерації. Наведено приклади застосування запропонованого узагальненого методу для генерації деяких класичних та некласичних комбінаторних множин. Оцінено обчислювальну складність запропонованого алгоритму.

В дисертаційній роботі запропоновано методи повної та випадкової часткової генерації композиційних k-образів комбінаторних множин (k-множин) на основі узагальненого методу комбінаторної генерації, які складають основу методів комбінаторної оптимізації на k-множинах. Наведено приклади роботи запропонованих алгоритмів, оцінено їх обчислювальну складність.

У роботі введено нову комбінаторну множину – перестановки з частково заданою сигнатурою, що є узагальненням перестановок з заданими підйомами та спусками. Розв`язано задачі перечислення та генерації даного класу перестановок на базі узагальненого методу.

Розроблено математичні моделі задачі вивозу і доставки з тривимірними обмеженнями завантаження (PDPBS) та задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції (TYSP2), що використовують апарат комбінаторних конфігурацій замість традиційних

булевих змінних, що дозволяє знизити розмірність, урахувати додаткові властивості та підвищити ефективність розв`язання задач зазначених класів.

Отримали подальший розвиток стратегії та методи розв’язання зазначених задач перевезення та обробки вантажів завдяки використанню методів комбінаторної генерації і врахуванню додаткових властивостей задач, що дозволяє підвищити ефективність їх розв’язання. Зокрема, запропоновано постановку задачі вивозу і доставки (Pickup and Delivery Problem, PDP), що одночасно враховує послідовність завантаження тривимірних контейнерів та порядок відвідування пунктів вивозу і доставки, виключаючи можливість блокування одних контейнерів іншими під час розвантаження та забезпечуючи стійкість завантаження (PDP with 3D loading constraints, blocking prevention and stability check, PDPBS). При цьому при упаковці тривимірних вантажів у транспортний засіб враховується можливість їх повороту в горизонтальній площині, що дозволяє більш оптимально розміщувати вантажі. Наведено стратегію розв’язання задачі, що базується на використанні відомих методів кластеризації *k*-середніх, методів комбінаторної генерації, математичного апарату Ф-функцій, запропонованого Ю.Г. Стояном, та методів упаковки *n*-вимірних паралелепипедів. Для побудови допустимих маршрутів транспортних засобів використовується евристика променевого пошуку та узагальнений метод комбінаторної генерації. Стратегія розв’язання включає етапи розподілу пар pickup-delivery між транспортними засобами та формування оптимальних маршрутів для кожного транспортного засобу. Метод формування оптимальних маршрутів полягає в генерації допустимих маршрутів і виборі кращого з них. Для цього використовується узагальнений метод та класична евристика променевого пошуку (beam search). Перевагою наведеного методу є можливість балансування між часом його роботи та точністю отриманих результатів, що дозволяє застосовувати описаний метод як в системах реального часу, де швидкість розв’язання є вирішальним фактором, так і в системах, де більш важливою є точність отриманого розв’язку.

Розроблено математичну модель та розвинуто стратегії та методи розв’язання задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції (Transshipment Yards Scheduling Problem, TYSP). Розроблена математична модель є більш детальною, ніж у існуючих роботах, додатково враховуючи призначення поїздів на залізничні колії, що дозволяє знизити вартість функціонування сортувальної станції. Для опису задачі використовуються комбінаторні конфігурації на відміну від існуючих робіт, що використовують булеві змінні. Множина допустимих розв’язків розглядуваної задачі описується за допомогою композиційного *k*-образу комбінаторних множин – перестановок розміщень.

Запропонований евристичний метод розв’язання задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції є подальшим розвитком методу розв’язання, що використовувався в існуючих роботах, присвячений цій задачі. Метод ґрунтується на застосуванні евристики променевого пошуку та узагальненого методу для генерації допустимих розкладів.

На основі методів та алгоритмів розв’язання описаних в роботі задач комбінаторної генерації та оптимізації розроблено програмний комплекс, що складається з модулів, кожен з яких розв’язує конкретну задачу комбінаторної генерації або оптимізації. Всі модулі мають дружній інтерфейс, а деякі з них доступні онлайн.

Для досліджуваних задач комбінаторної оптимізації проведено серію обчислювальних експериментів, на основі яких зроблено аналіз отриманих результатів та їх порівняння з результатами, отриманими в інших дослідженнях.

Практичне значення результатів роботи підтверджується актами впровадження: методів генерації комбінаторних конфігурацій, математичних моделей задач перевезення та обробки вантажів – в держбюджетні наукові дослідження; програмного забезпечення для розв’язання задачі вивозу та доставки вантажів і задачі складання розкладу руху вантажних поїздів та

обробки вантажів на сортувальній станції – в ТОВ «Клауд Воркс» для розв'язання задач, пов'язаних з перевезенням та обробкою вантажів (Додаток Б).

Запропоновані методи та програмне забезпечення для розв'язання задач вивозу і доставки та складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції можуть бути використані для підвищення ефективності діяльності транспортних компаній.

Матеріали дисертації досить повно викладено у 16 роботах: з них них 3 статті в спеціалізованих виданнях, затверджених ДАК МОН України, 4 статті у закордонних виданнях (всі праці входять до міжнародних наукометричних баз) та 9 тез доповідей на міжнародних конференціях.

Автор має 4 роботи (2 статті та 2 тези доповідей), що індекусються базою SCOPUS та 1 статтю, що індексується базою Web of Science.

Ключові слова: комбінаторна генерація, комбінаторна оптимізація, евристика, променевий пошук, k-множини, перестановки з заданими підйомами та спусками, задача маршрутизації транспорту, інтермодальні перевезення.

Список публікацій здобувача

1. И. В. Гребенник, А. С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Кибернетика и системный анализ. 2012. № 6 (48). С. 96-105.

2. И.В. Гребенник, О.С. Титова, А.С. Литвиненко. Оптимизация линейной функции на множестве циклических перестановок. Бионика интеллекта. 2012. № 2 (67). С. 8-12.

3. Grebennik I., Lytvynenko O. Random generation of combinatorial sets with special properties. Econtechmod: an international quarterly journal on economics of technology and modelling processes. 2016. vol. 5, no. 4. pp. 43–48.

4. Yuri G. Stoyan, Igor V. Grebennik, Viacheslav V. Kalashnikov, Oleksandr S. Lytvynenko. Enumeration and Generation of Permutations with a

Partially Fixed Order of Elements. International Journal of Combinatorial Optimization Problems and Informatics. 2017. Vol. 8, No. 1. pp. 19-30.

5. Rémy Dupas, Igor Grebennik, Oleksandr Lytvynenko, Oleksij Baranov. An Heuristic Approach to Solving the one-to-one Pickup and Delivery Problem with Three-dimensional Loading Constraints. International Journal of Information Technology and Computer Science(IJITCS). 2017. Vol.9, No.10. pp.1-12.

6. Igor Grebennik, Rémy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Scheduling Freight Trains in Rail-rail Transshipment Yards with Train Arrangements. International Journal of Intelligent Systems and Applications (IJISA). 2017. Vol.9, No.10, pp.12-19.

7. И. В. Гребенник, А. С. Литвиненко. Генерация перестановок с частично определенным порядком следования элементов. Бионика интеллекта. 2017. №2 (89). С. 26–30.

8. И.В. Гребенник, А.С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Материалы конференции «XIX International Conference ‘Problems of decision making under uncertainties’». 23-27 апреля 2012. Мукачево, Украина. С. 88-89.

9. I.V. Grebennik, O.S. Lytvynenko, O.S. Titova. Optimization of linear functions on cyclic permutations. Материалы конференции «XX International Conference ‘Problems of decision making under uncertainties’». 17-21 сентября 2012. Брно, Чехия. С. 43-44.

10. I. Grebennik, O. Lytvynenko. Random and exhaustive generation of combinatorial sets with special properties. Материалы конференции «5th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2014)». 24-25 ноября 2014. University of National and World Economy (UNWE). София, Болгария. С. 170-177.

11. R. Dupas, I. Grebennik, O. Lytvynenko. Combinatorial mathematical model and decision strategy for one-to-one pickup and delivery problem with 3D loading constraints. Материалы конференции «X International Scientific and

Technical Conference Computer Sciences and Information Technologies (CSIT 2015)». 14-17 сентября 2015. Lviv Polytechnic National University. Львов, Украина. С. 133-135.

12. R. Dupas, I. Grebennik, O. Lytvynenko. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «V Konferencja Zastosowań Matematyki w Technice, Informatyce i Ekonomii». 17 сентября 2015. Instytut Matematyki Wydział Matematyki Stosowanej Politechnika Śląska. Гливице, Польша. С. 26-31.

13. O. Lytvynenko, O. Baranov, R. Dupas, I. Grebennik. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «6th International Conference on Information Systems, Logistics and Supply Chain (ILS 2016)». 1-4 июня 2016. Бордо, Франция. С. 101-108.

14. И. В. Гребенник, А.С. Литвиненко. Перечисление и генерация перестановок с частично заданным порядком следования элементов. Материалы конференции «Сучасні проблеми машинобудування». 21-22 ноября 2016. Харьков, Украина. С. 26.

15. Igor Grebennik, Remy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Deciding on train arrangement in freight trains scheduling problem. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 35-41.

16. Igor Grebennik, Oleksandr Lytvynenko. Developing software for solving some combinatorial generation and optimization problems. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 58-64.

ABSTRACT

Lytvynenko O.S. Methods of generation of combinatorial configurations and their application in mathematical and computer modeling of freight transportation and processing problems. – Qualifying scientific work on the rights of the manuscript.

A Thesis for a Candidate of Engineering Sciences degree in the specialty 01.05.02 – mathematical modeling and computational methods.

Dissertation is written in Kharkiv National University of Radio Electronics, Ministry of Education and Science of Ukraine.

Thesis presentation will be held in A. Podgorny Institute for Mechanical Engineering Problems, NAS Ukraine, Kharkiv, 2019.

In the dissertation work methods of generation of classical and nonclassical combinatorial configurations are considered, as well as the application of these methods in mathematical and computer modeling of freight transportation and processing problems. The object of research in the work are the processes of generation of combinatorial configurations and mathematical and computer modeling of freight transportation and processing problems, the subject - the methods of generating combinatorial configurations, mathematical and computer simulations in solving freight transportation and processing problems.

Methods of research: methods of combinatorial analysis, combinatorial generation and optimization, geometrical design methods for constructing a mathematical model of the PDP problem, in particular, Φ-functions method for describing three-dimensional loading restrictions, heuristic methods for constructing and analyzing recursive trees during generation and combinatorial optimization.

The paper develops strategies and methods for generation of combinatorial configurations in terms of the use of recursive procedures and the generalization of generating combinatorial configurations. For this purpose, a strategy and a generalized method of combinatorial generation are developed to serve as the basis of other results in thesis. The method is based on the use of the recursive

backtracking procedure to form one or more elements of a combinatorial set of length i +1 from a partial element of length i. During its functioning, the method builds a backtracking tree. The universality of the proposed method is achieved due to the possibility of using various formal rules for constructing a concrete combinatorial set. A recursive algorithm *GenBase* is proposed, which implements a generalized combinatorial generation method. Examples of the application of the proposed generalized method for generating some classical and nonclassical combinatorial sets are given. The computational complexity of the proposed algorithm is evaluated.

In the dissertation the methods of full and random partial generation of composite k-images of combinatorial sets (k-sets) are proposed. These methods serve as the basis for combinatorial optimization methods on k-sets. The examples of the application of proposed algorithms are given, their computational complexity is evaluated as well.

The new combinatorial set is introduced – permutations with a partially given signature, which generalizes permutations with given ups and downs. The problems of enumeration and generation of this class of permutations on the basis of the generalized method are solved.

The mathematical models of Pickup and Delivery Problem with three-dimensional loading constraints (PDPBS) and Transshipment Yards Scheduling Problem problem (TYSP2) are developed. These models use combinatorial configurations instead of the traditional boolean variables, which allows to reduce the dimensionality of the model, take into account additional properties and improve the efficiency of solving the problems.

Further development of strategy and methods for solving mentioned freight transportation and processing problems is achieved through the use of combinatorial generation methods and the consideration of additional properties of problems, which allows to increase the efficiency of their solution. In particular, thesis describes formulation of the Pickup and Delivery Problem (PDP), which takes into account the sequence of loading of three-dimensional containers and the order of

visiting the pickup and delivery points, while preventing blocking one container by others during unloading and ensuring the loading stability (PDP with 3D loading constraints, blocking prevention and stability check, PDPBS). Also, during packing of three-dimensional freights in the vehicle, the possibility of their rotation in the horizontal plane is taken into account, which allows to pack the freights more optimal. The strategy for solving a problem is based on the use of well-known *k*-means clustering method, combinatorial generation methods, mathematical apparatus of Φ-functions, proposed by Yu. Stoyan, and packing methods of n-dimensional parallelepipeds. The beam search heuristic and the generalized method of combinatorial generation are used to construct transport routes. The developed solution strategy includes the steps of distributing of pickup-delivery pairs between vehicles and finding the optimal routes for each vehicle. The idea of the method of finding the optimal routes is to generate valid routes and choose the best of them. For this, we use the generalized generation method and the beam search heuristic. The advantage of the proposed solution method is the possibility of balancing between the computational time the results accuracy, which allows the described method to be used both in real-time systems, where the speed of the solution is a critical factor, and in systems where the solution accuracy is more important.

The mathematical model as well as strategies and methods for solving the Transshipment Yards Scheduling Problem (TYSP) were developed. The mathematical model is more detailed than in existing works, due to additionally taking into account the arrangement of trains on railways, which reduces the cost of functioning of the sorting station. Developed mathematical model uses combinatorial configurations in contrast to existing works which use Boolean variables. The feasible region of the considered problem is modeled by means of the *k*-set – permutations of arrangements.

The proposed heuristic method for solving the Transshipment Yards Scheduling Problem is the further development of the solution method used in existing works devoted to TYSP. The method is based on the application of beam search heuristic and the generalized method for generating the feasible region.

On the basis of methods and algorithms for solving the problems of combinatorial generation and optimization described in the thesis, a software consisting of modules, each of which solves a specific problem of combinatorial generation or optimization, has been developed. All modules have a user-friendly interface, and some of them are available online.

For the considered problems of combinatorial optimization, a series of computational experiments was conducted, on the basis of which the analysis of the results and their comparison with the results obtained in other studies.

The practical value of the results obtained in thesis is confirmed by the corresponding acts of implementation. Methods of generation of combinatorial configurations, mathematical models of freight transportation and processing problems were implemented in state budget scientific researches; software for solving the Pickup and Delivery Problem (PDP) and Transshipment Yards Scheduling Problem (TYSP) – in LLC "Cloud Works" to solve problems related to the transportation and processing of goods.

The developed methods and the software for the solution of PDP and TYSP can be used to improve the efficiency of functioning of transport companies.

The materials of the thesis are published in 16 works: 3 articles – in specialized journals approved by the Ministry of Education and Science of Ukraine, 4 articles – in world journals (all articles are indexed by international scientific databases) and 9 international conference theses. 4 works (2 articles and 2 conference theses) are indexed in SCOPUS; 1 article is indexed by Web of Science.

Keywords: combinatorial generation, combinatorial optimization, freight transportation and processing problems, heuristics, Pickup and Delivery Problem, beam search, k-sets, intermodal transportation.

List of publications of the applicant

1. И. В. Гребенник, А. С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Кибернетика и системный анализ. 2012. № 6 (48). С. 96-105.

2. И.В. Гребенник, О.С. Титова, А.С. Литвиненко. Оптимизация линейной функции на множестве циклических перестановок. Бионика интеллекта. 2012. № 2 (67). С. 8-12.

3. Grebennik I., Lytvynenko O. Random generation of combinatorial sets with special properties. Econtechmod: an international quarterly journal on economics of technology and modelling processes. 2016. vol. 5, no. 4. pp. 43–48.

4. Yuri G. Stoyan, Igor V. Grebennik, Viacheslav V. Kalashnikov, Oleksandr S. Lytvynenko. Enumeration and Generation of Permutations with a Partially Fixed Order of Elements. International Journal of Combinatorial Optimization Problems and Informatics. 2017. Vol. 8, No. 1. pp. 19-30.

5. Rémy Dupas, Igor Grebennik, Oleksandr Lytvynenko, Oleksij Baranov. An Heuristic Approach to Solving the one-to-one Pickup and Delivery Problem with Three-dimensional Loading Constraints. International Journal of Information Technology and Computer Science(IJITCS). 2017. Vol.9, No.10. pp.1-12.

6. Igor Grebennik, Rémy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Scheduling Freight Trains in Rail-rail Transshipment Yards with Train Arrangements. International Journal of Intelligent Systems and Applications (IJISA). 2017. Vol.9, No.10, pp.12-19.

7. И. В. Гребенник, А. С. Литвиненко. Генерация перестановок с частично определенным порядком следования элементов. Бионика интеллекта. 2017. №2 (89). С. 26–30.

8. И.В. Гребенник, А.С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Материалы конференции «XIX International Conference ‘Problems of decision making under uncertainties’». 23-27 апреля 2012. Мукачево, Украина. С. 88-89.

9. I.V. Grebennik, O.S. Lytvynenko, O.S. Titova. Optimization of linear functions on cyclic permutations. Материалы конференции «XX International Conference ‘Problems of decision making under uncertainties’». 17-21 сентября 2012. Брно, Чехия. С. 43-44.

10. I. Grebennik, O. Lytvynenko. Random and exhaustive generation of combinatorial sets with special properties. Материалы конференции «5th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2014)». 24-25 ноября 2014. University of National and World Economy (UNWE). София, Болгария. С. 170-177.

11. R. Dupas, I. Grebennik, O. Lytvynenko. Combinatorial mathematical model and decision strategy for one-to-one pickup and delivery problem with 3D loading constraints. Материалы конференции «X International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT 2015)». 14-17 сентября 2015. Lviv Polytechnic National University. Львов, Украина. С. 133-135.

12. R. Dupas, I. Grebennik, O. Lytvynenko. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «V Konferencja Zastosowań Matematyki w Technice, Informatyce i Ekonomii». 17 сентября 2015. Instytut Matematyki Wydział Matematyki Stosowanej Politechnika Śląska. Гливице, Польша. С. 26-31.

13. O. Lytvynenko, O. Baranov, R. Dupas, I. Grebennik. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «6th International Conference on Information Systems, Logistics and Supply Chain (ILS 2016)». 1-4 июня 2016. Бордо, Франция. С. 101-108.

14. И. В. Гребенник, А.С. Литвиненко. Перечисление и генерация перестановок с частично заданным порядком следования элементов. Материалы конференции «Сучасні проблеми машинобудування». 21-22 ноября 2016. Харьков, Украина. С. 26.

15. Igor Grebennik, Remy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Deciding on train arrangement in freight trains scheduling problem. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 35-41.

16. Igor Grebennik, Oleksandr Lytvynenko. Developing software for solving some combinatorial generation and optimization problems. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 58-64.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VRP – Vehicle Routing Problem

PDP – Pickup and Delivery Problem

PDPBS – Pickup and Delivery Problem with 3D loading constraints, blocking prevention and stability check

LIFO – last-in-first-out

FIFO – first-in-first-out

RBW – relative beam width

TYSP – Transshipment Yards Scheduling Problem

TYSP2 – Transshipment Yards Scheduling Problem, level 2

ВСТУП

Актуальність теми. Багато наукових та прикладних задач мають дискретну структуру і можуть бути описані за допомогою комбінаторних конфігурацій. Значна кількість задач цього класу може бути адекватно подана за допомогою комбінаторних оптимізаційних моделей. Зокрема, до них належать комбінаторні задачі розміщення геометричних об'єктів, задачі транспортної маршрутизації, задачі інтермодальних контейнерних перевезень, задачі про призначення тощо. Задачам комбінаторної оптимізації присвячено багато досліджень закордонних та вітчизняних вчених, серед яких N. Christofides, A. Schrijver, C.H. Papadimitriou, P. Toth, D. Vigo, P.M. Pardalos, S. Martello, І.В. Сергієнко, М.З. Згуровський, Ю.Г. Стоян, С.В. Яковлев, Н.В.Семенова [1]–[11]. В науковій школі Ю.Г. Стояна задачі комбінаторної оптимізації досліджувались в рамках напряму геометричного проектування в роботах Ю. Г. Стояна, С.В. Яковлева, М.І. Гіля, В.М. Комяк, О.О. Ємця, О.В. Панкратова, Т.Є. Романової, І.В. Гребенніка [9], [10], [12]–[15].

Серед проблем, які зводяться до задач комбінаторної оптимізації, останнім часом стають все більш актуальними задачі перевезення та обробки вантажів, в яких транспортування вантажів поєднується з їх різноманітним обробленням (упаковкою, перевантаженням, тимчасовим зберіганням тощо). Вони виникають, зокрема, у контейнерних інтермодальних перевезеннях, у транспортній маршрутизації, при плануванні робіт в морських портах, на вантажних залізничних станціях, великих підприємствах, товарних складах [4], [16]–[24].

Важливим засобом комбінаторики при дослідженні задач, що мають дискретну структуру, є комбінаторна генерація. Генерація різноманітних комбінаторних конфігурацій використовується при розробці та реалізації методів і алгоритмів розв’язання багатьох наукових та прикладних задач. Комбінаторній генерації присвячено багато досліджень, зокрема, роботи D. Knuth, F. Ruskey, D. Kreher, D. Stinson [25]–[28]. Методи комбінаторної

генерації використовуються в математичному і комп’ютерному моделюванні та аналізі об’єктів з дискретною структурою, під час розв’язання задач комбінаторної оптимізації та в інших областях [29]–[31]. Прикладами таких задач є задачі розподілу співробітників між проектами, задачі розділення культур між різними полями в агрономії, задачі проектування електричних ланцюгів, знаходження альтернативних шляхів транзакції коштів (банківська справа) та шляхів передачі повідомлень до віддалених супутників (космічна індустрія), задачі складання розкладу роботи цехів, задачі конфігурації обладнання на робочих місцях, задачі опису різноманітних зв'язків між атомами та молекулами, а також розміщення атомів всередині молекул [30]. Важливим є також застосування методів комбінаторної генерації в задачах обробки природної мови для опису різноманітних мовних шаблонів (паттернів), а також взагалі в розв’язанні задач прикладної лінгвістики [32]. Множини допустимих розв’язків багатьох задач можна описати за допомогою комбінаторних конфігурацій, а процес їх розв’язання описати як задачу комбінаторної оптимізації на деякій комбінаторній множині. Прикладами таких задач є задачі транспортної маршрутизації [4], [33], [34] та задачі геометричного проектування [9], [12], [35].

Методи комбінаторної генерації можуть бути застосовані в стратегіях розв’язання задач комбінаторної оптимізації, що використовують генерацію елементів області допустимих розв’язків. При цьому в реальних задачах виникає багато обмежень на допустимі розв'язки. Тому застосування класичних комбінаторних конфігурацій (перестановок, сполучень тощо) для опису та розв’язання таких задач є надлишковим.

Як наслідок, виникає необхідність у застосуванні некласичних комбінаторних конфігурацій, що враховують властивості та обмеження задач. Використання методів генерації некласичних комбінаторних конфігурацій для розв’язання задач комбінаторної оптимізації дозволяє генерувати одразу допустимі розв’язки, завдяки чому можна уникнути надлишковості та знизити обчислювальну складність методів комбінаторної оптимізації.

Серед некласичних комбінаторних конфігурацій, для формалізованого опису множини допустимих розв'язків задач, що мають складну комбінаторну структуру (наприклад [35]–[41]), може бути використаний апарат композиційних *k*-образів комбінаторних множин (*k*-множини), запропонований в роботах Ю.Г. Стояна та І.В. Гребенніка [42]–[44]. Не дивлячись на добре розроблений математичний апарат, задача генерації *k*-множин у загальній постановці залишається нерозв'язаною (розглянуто лише загальні принципи генерації [44] та один з її окремих випадків [45]).

Розв'язання задачі генерації *k*-множин, а також багатьох задач комбінаторної оптимізації, в яких множина допустимих розв'язків описується за допомогою комбінаторних множин, в свою чергу, вимагає розв'язання задач генерації базових комбінаторних множин.

Використання генерації різноманітних комбінаторних конфігурацій в методах розв'язання задач комбінаторної оптимізації вимагає побудови досить універсальних, гнучких методів комбінаторної генерації, що можуть враховувати обмеження таких задач. Незважаючи на велику кількість досліджень, присвячених генерації комбінаторних конфігурацій, більшість існуючих методів орієнтовані на генерацію вузьких класів комбінаторних множин і важко піддаються модифікації. Так, для багатьох класичних та некласичних комбінаторних множин описані алгоритми їх генерації (див., наприклад, роботи D. Knuth, F. Ruskey, D. Kreher, D. Stinson [25]–[28]), проте в більшості випадків кожен алгоритм генерації ґрунтується на специфічних властивостях конкретної комбінаторної множини. Тому виникає необхідність в узагальненому підході до генерації різноманітних базових комбінаторних конфігурацій, який можна було б використовувати як базу для розв'язання задач комбінаторної оптимізації в цілому.

Таким чином, *актуальними* є розробка та розвиток стратегій і методів генерації комбінаторних конфігурацій, зокрема некласичних, використання їх для математичного та комп'ютерного моделювання задач, що мають

дискретну структуру, та підвищення ефективності розв’язання задач комбінаторної оптимізації, зокрема, задач перевезення та обробки вантажів.

Мета та задачі дослідження. Метою дисертаційної роботи є розробка методів генерації комбінаторних конфігурацій та підвищення на їх основі ефективності розв’язання задач перевезення та обробки вантажів при їх математичному та комп’ютерному моделюванні.

Для досягнення цієї мети поставлено наступні задачі:

1. Розробка стратегії та узагальненого методу генерації комбінаторних конфігурацій як основи методів розв’язання задач комбінаторної оптимізації.
2. Розробка методу генерації композиційних *k*-образів комбінаторних множин (*k*-множин) з використанням запропонованого узагальненого методу.
3. Опис та розв’язання задач перечислення і генерації перестановок з частково заданою сигнатурою як узагальнення перестановок з заданими підйомами та спусками.
4. Побудова математичної моделі задачі вивозу і доставки (Pickup and Delivery Problem, PDP) на основі комбінаторних конфігурацій, що одночасно враховує послідовність завантаження тривимірних контейнерів та порядок відвідування пунктів вивозу і доставки, виключаючи можливість блокування одних контейнерів іншими під час розвантаження та забезпечуючи стійкість завантаження. Розробка методу на основі комбінаторної генерації, алгоритму та програмного забезпечення для розв’язання цієї задачі.
5. Побудова математичної моделі задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції з урахуванням призначення поїздів на колії на основі комбінаторних конфігурацій. Розробка методу на основі комбінаторної генерації, алгоритму та програмного забезпечення для розв’язання цієї задачі.

**Об`єкт дослідження** – процеси генерації комбінаторних конфігурацій та математичного і комп`ютерного моделювання задач перевезення та обробки вантажів, які мають комбінаторну структуру.

Предмет дослідження – методи генерації комбінаторних конфігурацій, математичне та комп'ютерне моделювання під час розв`язання задач перевезення та обробки вантажів.

Методи дослідження. В роботі застосовуються методи комбінаторного аналізу, комбінаторної генерації та оптимізації, методи геометричного проектування для побудови математичної моделі задачі вивозу і доставки (Pickup and Delivery Problem, PDP), зокрема, метод Ф-функцій для опису тривимірних обмежень завантаження, евристичні методи побудови та аналізу рекурсивних дерев при генерації та комбінаторній оптимізації.

Наукова новизна отриманих результатів. Наукова новизна дисертаційної роботи полягає в наступному:

1. Отримали подальший розвиток стратегії та методи генерації комбінаторних конфігурацій у частині використання рекурсивних процедур та узагальнення класів комбінаторних конфігурацій, які генеруються, що дозволяє підвищити універсальність генерації та розширити класи генерованих комбінаторних множин під час математичного та комп`ютерного моделювання задач, які мають комбінаторну структуру.

2. Вперше розроблено методи повної та випадкової часткової генерації композиційних *k*-образів комбінаторних множин (*k*-множин) на основі розробленого узагальненого методу комбінаторної генерації, які складають основу методів комбінаторної оптимізації на *k*-множинах.

3. Введено нову комбінаторну множину – перестановки з частково заданою сигнатурою, що є узагальненням перестановок з заданими підйомами та спусками, для якої розв`язано задачі перечислення та генерації на базі запропонованого методу.

4. Вперше розроблено математичні моделі задачі вивозу і доставки з тривимірними обмеженнями завантаження (PDPBS) і задачі складання

розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції, шо використовують апарат комбінаторних конфігурацій замість традиційних булевих змінних, що дозволяє знизити розмірність, урахувати додаткові властивості та підвищити ефективність розв`язання задач зазначених класів.

5. Отримали подальший розвиток стратегії та методи розв'язання задач перевезення та обробки вантажів завдяки використанню методів комбінаторної генерації і врахуванню додаткових властивостей задач, що дозволяє підвищити ефективність їх розв'язання, зокрема:

- задачі PDPBS в частині одночасного врахування послідовності завантаження тривимірних контейнерів, порядку відвідування пунктів вивозу і доставки, виключення можливості блокування одних контейнерів іншими під час розвантаження та забезпечення стійкості завантаження, регулювання балансу між часом на отримання розв'язку та точністю отриманих результатів за допомогою евристики променевого пошуку;
- задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції в частині врахування призначення поїздів на залізничні колії.

Особистий внесок здобувача. В роботах [46]–[49] автору належать метод генерації базових множин та алгоритм *GenBase,* метод генерації *k*-множин та алгоритм *Gen_k-set,* а також усе програмне забезпечення; в роботах [50]–[52] – математичний опис, метод перечислення, а також метод, алгоритм та програмне забезпечення для генерації; в роботах [40], [53]–[55] – математична модель досліджуваної задачі PDP, а також метод, алгоритм та програмне забезпечення для розв'язання описуваної задачі; в роботах [41], [56] – комбінаторні аспекти математичної моделі, методу, алгоритму та програмне забезпечення для розв'язання задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції; в роботі [57] – створене програмне забезпечення; в роботах [58], [59] – алгоритм пошуку евристичного розв'язання задачі.

Апробація результатів дисертації. Основні положення дисертаційної роботи було представлено на:

- 19-й та 20-й міжнародних конференціях «Problems of decision making under uncertainties» (Мукачево, Україна та Брно, Чехія, 2012 р.);
- 5-й та 7-й міжнародних конференціях «International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE)» (Софія, Болгарія, 2014 та 2017 рр.);
- 5-й міжнародній конференції «Konferencja Zastosowań Matematyki w Technice, Informatyce i Ekonomii» (Глівіце, Польша, 2015 р.)
- 10-й міжнародній конференції «Computer Sciences and Information Technologies (CSIT)» (Львів, Україна, 2015 р.) – індексується у міжнародній базі Scopus
- 6-й міжнародній конференції «International Conference on Information Systems, Logistics and Supply Chain (ILS)» (Бордо, Франція, 2016 р.) – індексується у міжнародній базі Scopus
- конференції молодих вчених та спеціалістів «Сучасні проблеми машинобудування» (Київ, Україна, 2017 р.)
- семінарах відділу математичного моделювання і оптимального проектування Інституту проблем машинобудування ім. А. М. Підгорного НАН України (м. Харків, 2018 р.).

Зв'язок роботи з науковими програмами, планами, темами. Робота виконана в період з 2014 по 2018 рр. на кафедрі системотехніки Харківського національного університету радіоелектроніки в рамках розділу №293-2 «Генерація комбінаторних конфігурацій в задачах математичного моделювання стійкого розвитку соціально-економічних систем» науково-дослідної роботи «Розробка методології і математичних моделей соціально-економічних систем при реалізації концепції їх стійкого розвитку» (ДР № 0115U001522), в розробці якої автор брав участь як виконавець.

В рамках робіт, що виконувались відповідно до плану НДР, автором проведено дослідження методів комбінаторної генерації та оптимізації, розроблено базовий метод генерації комбінаторних конфігурацій, розроблено методи повної та випадкової часткової генерації композиційних k-образів комбінаторних множин (k-множин), запропоновано нову комбінаторну множину – перестановки з частково заданою сигнатурою, розроблено математичні моделі задачі транспортної маршрутизації (Pickup and Delivery Problem з тривимірними обмеженнями завантаження) та задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції, що використовують апарат комбінаторних конфігурацій замість традиційних булевих змінних, а також методи їх розв'язання.

Практичне значення отриманих результатів. Практичне значення результатів роботи підтверджується актами впровадження: методів генерації комбінаторних конфігурацій, математичних моделей задач перевезення та обробки вантажів – в держбюджетні наукові дослідження; програмного забезпечення для розв'язання задачі вивозу та доставки вантажів і задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції – в ТОВ «Клауд Воркс» для розв'язання задач, пов'язаних з перевезенням та обробкою вантажів (Додаток Б).

Запропоновані методи та програмне забезпечення для розв'язання задач вивозу і доставки та складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції можуть бути використані для підвищення ефективності діяльності транспортних компаній.

Публікації. Матеріали дисертації досить повно викладено у 16 роботах: з них них 3 статті в спеціалізованих виданнях, затверджених ДАК МОН України [46], [50], [58], 4 статті у закордонних виданнях [40], [41], [47], [51] (всі праці входять до міжнародних наукометричних баз) та 9 тез доповідей на міжнародних конференціях [48], [49], [52]–[57], [59], 2 з яких індексуються у базі SCOPUS.

Автор має 4 роботи (2 статті [41], [46] та 2 тези доповідей [54], [55]), що індекусються базою SCOPUS та 1 статтю, що індексується базою Web of Science [51].

1 МЕТОДИ КОМБІНАТОРНОЇ ГЕНЕРАЦІЇ І РОЗВ'ЯЗАННЯ ЗАДАЧ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ

Класичні засоби комбінаторики застосовуються для математичного та комп'ютерного моделювання і розв'язання різноманітних наукових та прикладних задач. Математичне моделювання – це процес встановлення відповідності даному реальному об'єкту деякого математичного об'єкту, що називається математичною моделлю, та дослідження цієї моделі, що дозволяє отримати характеристики розглядуваного реального об'єкту [60]. Комп'ютерне моделювання – це метод розв'язання задачі аналізу або синтезу складної системи, що ґрунтується на використанні її комп'ютерної моделі. Під комп'ютерною моделлю розуміють умовний образ об'єкта чи деякої системи об'єктів або процесів, описаних за допомогою взаємозалежних комп'ютерних таблиць, схем, діаграм, графіків, малюнків, анімаційних фрагментів, гіпертекстів і т.д., що відбивають структуру та взаємозв'язки між елементами об'єкта чи системи [61].

В основі математичних моделей комбінаторних задач лежить поняття комбінаторної конфігурації, яке було формалізовано К. Бержем таким чином [62], [63]. Нехай задано натуральні числа m та n, а також множини $U = \{1, \ldots, m\}$, $V = \{v_1, \ldots, v_n\}$, причому на V задано деякий порядок $v_1 < \ldots < v_n$. Тоді комбінаторною конфігурацією є відображення $\phi: U \to V$, що задовольняє деякий комплекс обмежень Λ [62], [63]. При фіксованих m та n число комбінаторніх конфігурацій є кінцевим. Вибір обмежень Λ дозволяє описувати різноманітні комбінаторні конфігурації (перестановки, розміщення, сполучення і тому подібне). В роботі С.В. Яковлева та О.С. Пічугіної наводиться наступне визначення комбінаторного об'єкта [64]. Нехай $J_n = \{1,2,...,n\}, \mathbb{Z}$ – не більш ніж зліченна множина ізольованих точок, а Z – утворена нею базова множина. Тоді комбінаторним об'єктом є тріада (ψ, Z, Ω)

, де $\psi : J_n \to Z$ – гомоморфізм, що задовольняє деякій системі обмежень Ω; множина J_n зветься нумеруючою.

До класичних комбінаторних задач відносять наступні задачі [28].

Задача *генерації* полягає в побудові всіх комбінаторних структур визначеного типу [28]. Комбінаторній генерації присвячено багато робіт, наприклад, роботи Д. Крехера та Д. Стінсона, Д. Кнута та Ф. Раскі [25]–[28]. Алгоритми комбінаторної генерації дозволяють отримати всі комбінаторні об'єкти заданого типу (наприклад, перестановки, сполучення, розміщення тощо) в деякому порядку, наприклад, в лексикографічному.

Задача *перечислення* полягає в обчисленні загальної кількості комбінаторних структур заданого типу [28]. Кожен алгоритм комбінаторної генерації може бути використаний для перечислення згенерованих комбінаторних об'єктів в кінці своєї роботи. Але часто виникає необхідність наперед знати кількість усіх комбінаторних об'єктів заданого типу. Окрім того, перечислення комбінаторних об'єктів за відомою формулою можна провести значно швидше, ніж генерацію всіх об'єктів. Задачам перечислення комбінаторних об'єктів присвячено багато досліджень, наприклад, роботи Р. Стенлі, М. Бона, Д. Крехера та Д.Стінсона [28], [65], [66].

Важливою задачею комбінаторики в цілому є також задача *пошуку* хоча б одного комбінаторного об'єкту заданого типу. Типовим прикладом такої задачі є задача пошуку кліки (clique) заданого розміру на неорієнтованому графі [67]–[69], що може бути розв'язана, наприклад, алгоритмом Брона-Кербоша [69]. Для пошуку інколи можуть використовуватися алгоритми комбінаторної генерації, але це не завжди є ефективним методом.

Варіацією задач пошуку є задачі комбінаторної оптимізації, де потрібно знайти оптимальний (за деяким критерієм) комбінаторний об'єкт заданого типу. Оптимальність того чи іншого комбінаторного об'єкту визначається цільовою функцією, що може виражати, наприклад, якість отриманого розв'язання чи його вартість.

Комбінаторна оптимізація є важливим напрямком прикладної математики, що здобула широке застосування для розв'язання багатьох прикладних задач. Комбінаторній оптимізації присвячено багато досліджень як відчизняних [7]–[9], [11], [12], [14], [43], [63], [70]–[90], так і закордонних авторів (напр., [1]–[3], [5], [91], [92]).

Задачами комбінаторної оптимізації є, наприклад, транспортні задачі, задачі упаковки і розміщення об'єктів, задачі про призначення, задачі розкрою, задачі побудови мінімального остовного дерева.

Для аналізу цих задач використовуються математичні моделі у вигляді задач дискретної оптимізації і методи їх розв'язання.

1.1 Огляд методів комбінаторної оптимізації

Опишемо загальну постановку задачі комбінаторної оптимізації [7]. Нехай A – деяка кінцева множина, що зветься множиною твірних елементів. Позначимо через $X = X_A$ комбінаторний простір – сукупність всіх комбінаторних об'єктів, що утворені з елементів множини A. Кожній точці такого простору $x \in X$ можна поставити у відповідність деяку метрику $f(x)$, що зветься цільовою функцією. Тоді задачу комбінаторної оптимізації можна описати як задачу знаходження точки $x^* \in X$, такої, що цільова фунція $f(x)$ досягає екстремуму

$$x^* = \arg \min_{x \in X} f(x).$$

Прикладом задачі комбінаторної оптимізації є класична проблема комівояжера [93], [94], де множина A відповідає всім пунктам, шо треба обійти, комбінаторний простір – це множина перестановок елементів A (кожен маршрут описується перестановкою), а цільовою функцією $f(x)$ є вартість маршруту.

Розглянемо класифікацію методів комбінаторної оптимізації [81]. Поширеною класифікацією методів комбінаторної оптимізації є класифікація

за типом отримуваного розв’язку; за такою класифікацією виділяють точні, наближені та евристичні алгоритми.

Точні алгоритми за кінцевий час гарантовано дають оптимальний розв’язок або роблять висновок, шо його не існує, якщо задачу не можна розв’язати [81]. Серед точних методів розв’язання задач комбінаторної оптимізації можна виділити узагальнені методи, що можуть бути застосовані для розв’язання широкого класу задач, наприклад, метод повного перебору, метод гілок та меж, метод гілок і перетинів, динамічне програмування, аналіз варіантів і відсіювання результатів [3], [7], [95], а також спеціальні методи, що будуються з урахуванням специфіки конкретної задачі.

Наближені алгоритми за заданий кінцевий час дозволяють гарантовано отримати варіанти розв’язку задачі, точність яких може бути оцінена апріорно або апостеріорно.

Найбільшого розвитку останнім часом здобувають евристичні методи комбінаторної оптимізації. На відміну від точних та наближених методів, евристичні методи не дозволяють отримати оцінку точності результатів. Проте в реальній практиці ці методи демонструють свою ефективність, особливо для розв’язання таких задач, де застосування точних або наближених методів неможливе або потребує забагато часу. Часто, евристики можуть бути єдиним способом отримати «гарний» розв’язок за прийнятний час [81].

До евристик відносять «жадібні» (greedy) методи [96, pp. 67–80], локальний пошук [97], методи імітації відпалювання (simulated annealing) [99], *G*-алгоритм [100], пошук з табу [101], метод околів, що звужуються [9], генетичні алгоритми [102], меметичні алгоритми [103], оптимізацію мурав’їними колоніями [104], [105] і багато інших.

Однією з важливих технік, що застосовується у розв’язанні задач комбінаторної оптимзації, є занурення комбінаторних множин у евклідовий простір [3], [9], [10], [92], [106]–[108]. Зокрема, опукла оболонка множини перестановок являє собою перестановочний багатогранник, множина вершин якого відповідає множині перестановок [106]. Дана властивість

перестановочного багатогранника дає можливість звести задачу, визначену на дискретній комбінаторній множині, до задачі, що має безперервну множину допустимих розв'язків. Це дає можливість застосовувати класичні методи безперервної оптимізації для розв'язання задач комбінаторної оптимізації, а також розвивати нові методи розв'язання, використовуючи властивості комбінаторних множин і їх опуклих оболонок [107].

Для багатьох задач комбінаторної оптимізації множину допустимих розв'язків можна описати за допомогою комбінаторних конфігурацій: перестановок, сполучень, розміщень, розбивок тощо. Тому ключову для комбінаторної оптимізації роль грає генерація комбінаторних конфігурацій.

1.2 Комбінаторна генерація

Історичний огляд розвитку комбінаторної генерації дає Дональд Кнут в 4 томі «Мистецтва програмування» [25], [26]. Однак, класифікація алгоритмів генерації і визначення основних понять в роботі Кнута не наведені. Таку класифікацію приводить Френк Раскі в [27], де виділяється чотири класи алгоритмів генерації:

1) алгоритми послідовної генерації всіх елементів комбінаторної множини (listing algorithms);
2) алгоритми нумерації всіх елементів комбінаторних множин (ranking algorithms);
3) алгоритми генерації елементів комбінаторної множини відповідно заданій нумерації (unranking algorithms);
4) алгоритми випадкової генерації елементів комбінаторної множини (random selection algorithms).

Алгоритми послідовної генерації комбінаторних множин є найбільш вивченими. Принцип їх роботи полягає в наступному [28], [109], [110]:

1) ініціалізація початкового елемента комбінаторної множини;

2) організація циклу, в якому з поточного елементу комбінаторної множини виводиться наступний.

Цей процес можна реалізувати двома способами. Перший спосіб передбачає реалізацію процедури генерації як контейнера, тобто для всіх елементів даної комбінаторної множини викликається деяка задана процедура. Зазвичай такий механізм реалізується у вигляді процедури під назвою *ForEach*.

Інший спосіб полягає в реалізації процедур генерації *First* та *Next*. Процедура *First* забезпечує ініціалізацію початкового елемента. Процедура *Next* генерує наступний елемент.

Нумерація розглядається як процес упорядкування множини комбінаторних об'єктів. Якщо деяка множина комбінаторних об'єктів є впорядкованою, то позиція конкретного об'єкту буде його порядковим номером (rank). Відповідний алгоритм або процедуру зазвичай називають *Rank*.

Процес генерації комбінаторного об'єкта за його номером (*unranking*) є зворотним процесу нумерації. Відповідний алгоритм або процедуру називають *Generate*.

У деяких випадках буває необхідно згенерувати невпорядковану послідовність комбінаторних об’єктів, і немає потреби отримувати всю множину разом. Тоді використовують алгоритми генерації з випадковим вибором (random selection).

Методи і алгоритми комбінаторної генерації набувають важливе наукове і практичне значення, зокрема, в проектуванні інформаційних систем і баз даних [111], в теорії кодування і стиснення інформації [112], [113], в хімії при дослідженні різних хімічних структур [114], [115], в біології при моделюванні ДНК-структур [116], [117].

У загальному випадку, для побудови процедур генерації і перечислення комбінаторної множини необхідно задати бієктивне відображення [28]

$$Generate: N_m \to A_m,$$

де N_m – кінцева підмножина натуральних чисел;

A_m – результуюча комбінаторна множина.

Тоді зворотне відображення задає процедуру нумерації

$$Rank: A_m \to N_m.$$

Методи генерації і перечислення комбінаторних множин досить різноманітні і залежать від конкретної комбінаторної множини. Існують також комбінаторні методи перечислення та пошуку, що претендують на деяку універсальність застосування.

Так, Е.Рейнгольд, Ю.Нівергельт та Н.Део для розв'язання задач пошуку комбінаторних об'єктів запропонували використовувати метод, ґрунтується на пошуку з поверненням (backtracking) [118]. Подальший розвиток цього методу запропоновано Д.Крехером і Д.Стінсоном в [28]. Однак у існуючій літературі, присвяченій цьому методу, не було знайдено прикладів використання цього методу для генерації комбінаторних множин.

Одним з методів перечислення комбінаторних конфігурацій, який претендує на універсальність, є метод ECO (Enumeration Combinatorial Object), запропонований в [119], [120], в основі якого лежить використання рекурсивних правил:

1) комбінаторні об'єкти розмірності n+1 будуються з об'єкта розміру n;

2) кожен об'єкт розмірності n+1 має один батьківський об'єкт розміру n.

За цими правилами будується генеруюче дерево, в кожному вузлі якого записано число об'єктів. Даний метод добре зарекомендував себе для перечислення багатьох класів комбінаторних об'єктів [121]. Для генерації комбінаторних об'єктів використовується ідея переходу від об'єкта розмірності n до об'єкта розмірності n+1. Прикладом може служити алгоритм послідовної генерації деяких класів поліміно [122]. Число вузлів в дереві

пропорційно числу комбінаторних об'єктів розглянутого класу. В існуючій літературі не було знайдено прикладів використання даного методу для розв’язання задач комбінаторної генерації.

Слід зазначити також метод, запропонований [123] для побудови алгоритмів генерації елементів комбінаторних множин з випадковим вибором, що був розвинутий та застосований в [124]–[126] для побудови алгоритмів послідовної генерації, нумерації і генерації за номером. Цей метод базується на побудові відповідностей між твірними функціями, що описують потужності комбінаторних множин, і деякими операціями, визначеними над цими множинами. Крім відомих операцій додавання і множення, в методі на множинах введено спеціальні операції *Seq, Set, PSet, Circle*. Якщо множина A є кінцевою для всіх множин $Seq(A)$, то можна задати алгоритм комбінаторної генерації цієї множини. Основним недоліком даного методу є неможливість скористатися методом для побудови алгоритмів комбінаторної генерації, якщо твірна функція невідома; а коли вона відома, для використання методу її треба висловити в термінах операцій додавання, множення, *Seq, Set* та *Pset*, що є досить складним завданням.

Для нумерації елементів комбінаторних множин запропоновано метод, що ґрунтується на розгляданні комбінаторного об'єкту у вигляді бітової послідовності [112], [127], однак отримання бітового представлення елементів комбінаторної множини не наводиться.

У вказаних джерелах головним чином розв’язуються задачі генерації досить простих комбінаторних об'єктів – перестановок, сполучень, розбивок, дерев, двійкових послідовностей.

Однак, багато задач не можуть бути описані в термінах класичних комбінаторних множин, тому у багатьох випадках вирішення наукових та прикладних задач потребує розв’язання задач генерації некласичних комбінаторних множин, або нових з точки зору їх комбінаторної структури, або володіючих додатковими властивостями порівняно з класичними.

Прикладами таких множин є *k*-множини та перестановки з заданими підйомами та спусками.

1.3 Конструктивні засоби опису композиційних *k*-образів комбінаторних множин (*k*-множини)

Існує досить велика кількість практичних задач, що мають складну комбінаторну структуру, і для формалізованого опису множини допустимих розв'язків таких задач необхідно використовувати комбінаторні множини, що також мають відповідну структуру.

Досить складні комбінаторні конфігурації можуть бути формально описані і згенеровані за допомогою конструктивних засобів опису композиційних *k*-образів комбінаторних множин (*k*-множин), запропонованих в роботах Ю.Г. Стояна та І.В. Гребенніка [42]–[44].

Ідея композиційних *k*-образів комбінаторних множин (*k*-множин) полягає в описі складної комбінаторної конфігурації у вигляді ієрархічного дерева, вузлами якого є базові комбінаторні множини. При цьому під комбінаторною множиною розуміється множина кортежів, побудованих з кінцевої множини довільних елементів (так званих твірних елементів) відповідно до певних правил [44]. Прикладами базових комбінаторних множин можуть служити перестановки, сполучення, розміщення, двійкові послідовності. Основою методу побудови *k*-множин виступає операція *n*-заміщення, яка полягає в тому, що на кожному рівні ієрархії твірні елементи батьківської множини замінюються на елементи-кортежі, що входять до дочірніх множин.

Приклад. Нехай *k*-множина представлена деревом, наведеним на рис.1.1.

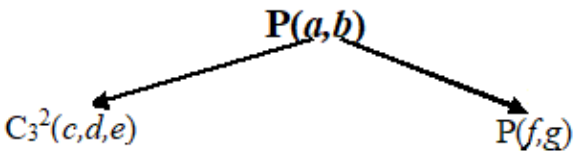


Рисунок 1.1. Приклад *k*-множини

Дерево на рис. 1.1 має два рівні ієрархії (нульовий та перший). На нульовому (верхньому) рівні знаходиться множина перестановок, породжених твірними елементами $\{a,b\}$, на першому (нижньому) рівні з лівої сторони – множина сполучень з трьох твірних елементів $\{c,d,e\}$ по два, з правої сторони – множина перестановок з елементів $\{f,g\}$.

Батьківська множина перестановок на нульовому рівні має 2 кортежі (*ab*) та (*ba*). При виконанні операції *n*-заміщення твірний елемент *a* батьківської множини буде послідовно замінений на кожен з кортежів, що складають дочірню множину сполучень, тобто на кортежі (*cd*), (*ce*) та (*de*). Утворюючий елемент *b* буде послідовно замінений на кожен з кортежів дочірньої множини перестановок (*fg*) та (*gf*).

Таким чином, результатом побудови композиційного *k*-образу комбінаторних множин, представлених ієрархічним деревом на рис. 1.1, є множина кортежів (*cdfg*), (*cdgf*), (*cefg*), (*cegf*), (*defg*) та (*degf*).

В роботі [15] запропоновано спеціальну термінологію для деяких *k*-множин, що мають два рівні ієрархії. Наприклад, *k*-множина, де на верхньому, нульовому рівні знаходиться множина перестановок з *n* елементів, а на нижньому, першому рівні – *n* множин сполучень, називається перестановкою сполучень. Перелік відповідних позначень наведено в таблиці *1.1*.

Спеціальні терміни для деяких *k*-множин *Таблиця 1.1*

	Множина на нульовому рівні				
Множина на першому рівні		Перестановки	Сполучення	Розміщення	Кортеж
	Перестановки	Композиція перестановок	Сполучення перестановок	Розміщення перестановок	Кортеж перестановок
	Сполучення	Перестановка сполучень	Композиція сполучень	Розміщення сполучень	Кортеж сполучень
	Розміщення	Перестановка розміщень	Сполучення розміщень	Композиція розміщень	Кортеж розміщень
	Кортеж	Перестановка кортежів	Сполучення кортежів	Розміщення кортежів	Кортеж кортежів

Апарат композиційних *k*-образів комбінаторних множин дозволяє описати область допустимих розв’язків багатьох задач, що мають складну комбінаторну структуру, наприклад [35], [36], [38]–[41].

В роботах [35], [36], [38] за допомогою *k*-множин описуються варіанти упаковки множини *n*-вимірних паралелепіпедів з можливістю зміни їх ортогональної орієнтації в *n*-вимірному паралелепипеді. В роботі розглянуто евристичний метод розв'язання такої задачі, що ґрунтується на послідовно-одиночному розміщенні паралелепипедів. Розглядається також задача розміщення об'єктів довільної форми [35]. Задача розв’язується шляхом упаковки об'єктів в *n*-вимірні паралелепіпеди за критерієм мінімального обсягу. При цьому для моделювання розміщення геометричних об’єктів використовується апарат Ф-функцій. Цей апарат був вперше описаний Ю.Стояном в [128] та здобув подальшого застосування в задачах геометричного проектування та моделювання [129]–[131].

1.4 Перестановки з заданими підйомами та спусками

Перестановки з заданими підйомами та спусками були вперше розглянуті в [132], де визначається, шо перестановка з заданими підйомами та спусками, або з заданою сигнатурою $Q = (q_1,q_2,...,q_{n-1})$, де q_i дорівнює 1 або -1 – це така перестановка $P = \pi_1, \pi_2,..., \pi_n$ цілих чисел від 1 до n, що $q_i \cdot (\pi_{i+1} - \pi_i)$ є додатнім для всіх $1 \leqslant i \leqslant n-1$.

Перестановки з заданими підйомами та спусками здобули подальшої уваги: в [133]–[136] було виведено формули для перечислення таких перестановок, в [137] був приведений алгоритм с постійним середнім часом (constant average time, CAT) для генерації всіх перестановок, шо відповідають заданій сигнатурі $Q = (q_1,q_2,...,q_{n-1})$.

Також перестановкам з заданими підйомами та спусками відповідають перестановки з заданою множиною спуску [65, p. 38], де сигнатура

перестановки $w = w_1 w_2 \ldots w_n$ описується за допомогою множини спуску $D(w) = \{i : w_i > w_{i+1}\} \subseteq \{1,2, \ldots n-1\}$. В [65] також наведено формулу для перечислення перестановок, що відповідають заданій множині спуску $D(w)$.

Варто зазначити, що окремим випадком перестановок з заданими підйомами та спусками є альтернативні перестановки [138], [139], які описуються сигнатурою $Q = (1, -1, 1, -1,...)$ та унімодальні перестановки [140]–[143], для яких $Q = (-1, -1, .., -1, 1, 1,..., 1)$ або $Q = (1, 1, .., 1, -1, -1,..., -1)$.

1.5 Задачі маршрутизації транспорту

Одним з найпопулярніших напрямків застосування моделей і методів комбінаторної оптмізації є математичне моделювання та розробка стратегій розв’язання задач маршрутизації транспорту (vehicle routing problems, VRP). Це широкий клас задач цілочисельного програмування, якому останнім часом приділяється все більш уваги через його ключову роль в транспортній логістиці. Задачам маршрутизації транспорту присвячено багато книг, статей та монографій, зокрема, [33], [144]–[150].

В таких задачах для заданого парку транспортних засобів, що знаходяться в одному або декількох депо, треба побудувати маршрути до споживачів з урахуванням різних обмежень: на вантажопідйомність транспорту, на часові вікна (time windows), в які клієнти можуть обслуговуватися і тому подібне.

Приклад типової задачі VRP та можливого її розв’язання для m=5 приведено на рис. 1.2.

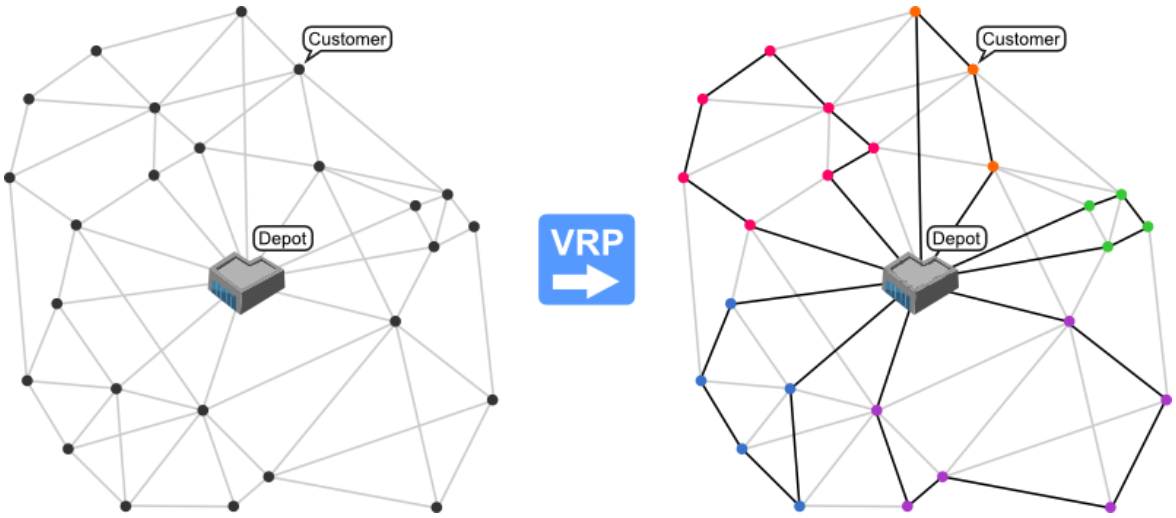


Рисунок 1.2. Приклад типової задачі VRP та можливого розв’язання для m=5

В реальних задачах виникає багато додаткових обмежень, яким відповідають підкласи задач VRP:

- Capacitated VRP (CVRP): кожний транспортний засіб має обмежену вантажопідйомність [151]–[153].
- VRP with Time Windows (VRPTW): кожен замовник може обслуговуватися лише в задане часове вікно [154]–[158].
- Multiple Depot VRP (MDVRP): для обслуговування клієнтів використовуються кілька депо [159]–[161].
- Split Delivery VRP (SDVRP): кожен клієнт може обслуговуватися декількома машинами [162]–[166].
- Periodic VRP (PVRP): доставка вантажів клієнту може здійснюватися протягом кількох днів [161], [167]–[171].
- Stochastic VRP (SVRP): деякі змінні (кількість і запити клієнтів, довжина шляху) можуть приймати випадкові значення [172]–[177].
- VRP with Satellite Facilities (VRPSF): існує можливість довантаження транспортного засобу на маршруті [178], [179].

Описані класи задач Vehicle Routing Problem стосуються різних практичних ситуацій, але зосереджуються на спільній проблемі – ефективному використанні парку транспортних засобів, що повинні обслуговувати замовлення клієнтів.

Одним з найпопулярніших класів задач маршрутизації транспорту є задачі вивозу і доставки (Vehicle Routing Problems with Pickups and Deliveries, VRPPD, або Pickup and Delivery Problem, PDP) [34], [180]–[182], де транспортні засоби обслуговують замовлення клієнтів (transportation requests), в кожному з яких треба забрати вантаж (або декілька вантажів) в пункті завантаження (pickup) та перевезти його в відповідний пункт доставки (delivery). Задача PDP виникає в багатьох контекстах, таких як міські кур'єрські послуги, системи транспортування від дверей до дверей та послуги таксі.

В задачах класу PDP маршрути транспортних засобів повинні бути прокладені таким чином, щоб кожен маршрут починався та закінчувався в депо та задовольняв обмеженням парності та пріоритету: для кожного запиту пункт завантаження (pickup) має передувати пункту доставки (delivery), і обидва пункти повинні бути відвідані одним і тим самим транспортним засобом. На рис. 1.3 приведено приклад задачі Pickup And Delivery Problem та її можливого розв’язання – маршруту транспортного засобу (депо позначено трикутником).

У випадку, коли перевозяться не вантажі, а люди, задача PDP називається задачею доставки по виклику (Dial-a-ride problem, DARP) [183], [184]. В такому випадку вага кожного «вантажу» приймається за одиницю, а вантажопідйомність транспортного засобу дорівнює кількості пасажирських місць.

Розглянемо класифікацію задач PDP.

За кількістю транспортних засобів бувають задачі PDP з 1 транспортним засобом (single vehicle PDP) або з декількома (multi vehicle PDP).

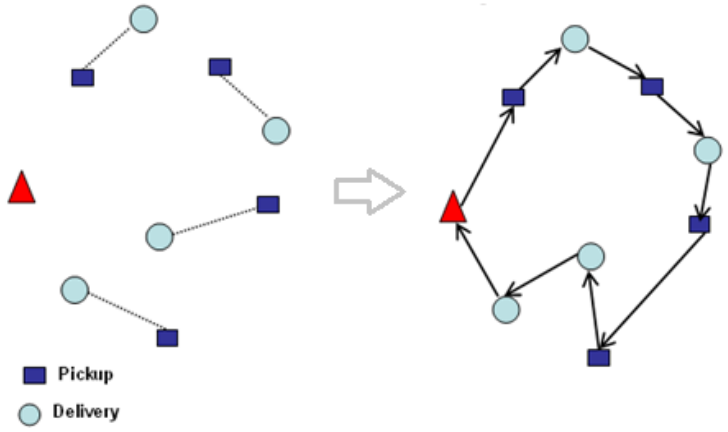


Рисунок 1.3. Приклад задачі PDP та можливого маршруту транспортного засобу

За динамічністю бувають статичні (static) та динамічні (dynamic) задачі PDP. В статичних задачах транспортні запроси є фіксованими [181], а в динамічних задачах в процесі обслуговування клієнтів можуть з'являтися нові транспортні запроси або змінюватися існуючі [185], [186].

Цільовою функцією в задачах PDP може бути [34]:

- Мінімізація тривалості маршруту. Тривалість маршруту включає час проїзду, очікування, завантаження та розвантаження, часи перерв. Мінімізація довжини маршруту, тобто сумарної відстані, що пройшли транспортні засоби.
- Мінімізація незручностей клієнтів. На незручність клієнтів впливає: відхилення бажаного та реального часу завантаження та розвантаження; різниця між часом їзди між пунктами завантаження та доставки навпростець (не заїжджаючи до інших клієнтів по дорозі) та реальним часом, за який вантаж було доставлено; різниця між часом появи транспортного запросу та прибуттям транспортного засобу до пункту завантаження (в динамічних задачах PDP)
- Мінімізація кількості транспортних засобів, що знадобилися для обслуговування клієнтів. Оскільки транспортні засоби та послуги водіїв є найдорожчою складовою для транспортних компаній, цей критерій може бути дуже важливим.

- Максимізація прибутку. Ця цільова функція може використовувати всі функції, наведені вище, для підрахунку «чистого» прибутку від обслуговування клієнта. В деяких транспортних системах диспетчер має можливість відхилити запит на перевезення, коли «чистий» прибуток невеликий (отриманий дохід не відшкодовує або мало відшкодовує витрати на перевезення). Варто зазначити, що в системах типу dial-a-ride (наприклад, таксі) відхилення запросів на перевезення неможливе або небажане.

Також в різних реальних задачах виникає багато додаткових обмежень.

Обмеження на часові вікна (time windows): кожен *i*-ий пункт завантаження (pickup) та доставки (delivery) має інтервал часу $[e_i, l_i]$, в який може здійснюватися обслуговування цього пункту. Ці обмеження є найбільш поширеними в математичних моделях задач PDP та часто враховуються в різноманітних задачах PDP [187]–[194].

Часові обмеження на режим роботи транспортних засобів (time constraints related to vehicles). Зазвичай транспортні засоби доступні не увесь час: водіям треба їсти і спати, а доступність вантажівок може регулюватися розкладом. Ці обмеження можна моделювати як вікна часу, протягом яких транспортні засоби є доступними [34, p. 10].

Важлимим для реальних задач класом обмежень є о*бмеження на завантаження транспортного засобу (loading contraints).* В найпростіших моделях враховується вантажопідйомність кожного транспортного засобу і маса кожного вантажу [195], [196].

В реальних задачах також дуже важливо забезпечувати безпроблемне діставання вантажу в пункті доставки: вантаж має стояти так, щоб його можна було легко дістати, і не повинен бути заблокований іншими вантажами. Щоб врахувати такі вимогу, можна накласти обмеження на порядок завантаження і розвантаження, описавши вантажне відділення транспортного засобу як стек «останнім прийшов, першим вийшов» (last-in-first-out, LIFO) [197], [198] або

як буфер «першим прийшов, першим вийшов» (first-in-first-out, FIFO) [193], [198], [199].

Більш адекватно порядок завантаження можна врахувати, якщо задати для кожного транспортного засобу розмір вантажного відділення, а для кожного вантажу, що треба доставити – його лінійні розміри. Тоді можна промоделювати реальне розташування вантажів всередині фури, і забезпечити таке завантаження, щоб при вивантаженні в пункті доставки предмет не був заблокований. Це означає, що в таких моделях, окрім задачі маршрутизації, треба розв'язати також задачу упаковки тривимірних об'єктів, що суттєво підвищує сумарну обчислювальну складність методу розв'язання задач з такими обмеженнями.

Розміри можуть бути задані двома вимірами (2D loading constraints), коли задається довжина і ширина вантажів та вантажних відсіків транспортних засобів, а їх висота вважається однаковою [200], [201], або трьома вимірами (3D loading constraints) [202], [203]. У останньому випадку вантажі можуть бути поставлені один зверху одного, і в таких випадках може також враховуватися крихкість (fragility) вантажів (на крихкі вантажі не можна нічого ставити зверху) та стійкість завантаження (вантажі, що погружені зверху, повинні стояти стійко).

Також під час розв'язання задач класу PDP важливим фактором, який треба враховувати, є можливість блокування одних контейнерів іншими під час розвантаження, що враховується, наприклад, в роботі [204], де блокування допускається, але при необхідності здійснюється перевантаження (reloading).

При цьому тривимірні обмеження завантаження (3D loading constraints) можуть бути описані за допомогою математичного апарату Ф-функцій, що використовувався в роботах Ю.Г.Стояна, М.І. Гіля та І.В. Гребенніка для формального опису умови взаємного перетину n-вимірних об'єктів [205]–

[207].

Різноманітні обмеження на завантаження транспортного засобу (в контексті задач маршрутизації взагалі) детально описані в оглядах [6], [208], [209].

1.6 Задачі інтермодальних перевезень

Інтермодальні перевезення можуть бути визначені як перевезення людей або вантажу від початкової точки до місця призначення щонайменше двома видами транспорту, перехід між якими здійснюється на інтермодальному терміналі [16]. Прикладом інтермодальних перевезень є транспортування контейнерних вантажів комбінацією вантажних, залізничних і морських транспортних засобів. Інтермодальні вантажні перевезення відносяться до мультимодального ланцюга контейнерно-транспортних послуг. Цей ланцюг зазвичай пов'язує первісного відправника з кінцевим одержувачем і тягнеться на великі відстані. Контейнерні перевезення є основним компонентом інтермодальних перевезень та міжнародної торгівлі [16].

Організованість інтермодальної системи істотно впливає на якість послуг, тому постійне впровадження новітніх інформаційних технологій є важливою умовою розвитку транспортних логістичних систем [210].

Основна задача, яку вирішують інтермодальні перевезення – це оптимізація часу та вартості пересування пасажирів або товарів, що досягається за рахунок пересадок пасажирів або перевантаження товірів з одного виду транспорту на інший. Також важливою метою, яку переслідують сучасні інтермодальні пасажирські перевезення, є мінімізація використання населенням персональних автомобілів для збільшення пасажиропотоку на публічний транспорт [210].

Інтермодальні перевезення також можна назвати новим етапом в області міжнародних перевезень вантажів. Такий вид перевезень забезпечує

підвищення надійності транспортного обслуговування, скорочення вартості і термінів доставки вантажів, запобігання затримки вантажу в пунктах перевантаження, а також управління всім процесом доставки вантажу зі складу відправника на склад одержувача одним оператором на основі єдиного технологічного графіку і новітніх комп'ютерних технологій [211].

Задачі моделювання та оптимізації інтермодальних перевезень здобули широкої уваги в сучасних наукових працях [16]–[18], [212]–[219].

Одним з важливих класів задач інтермодальних перевезень є задачі обробки контейнерів на залізничних сортувальних станціях (Container Processing in Railway Yards), що протягом останніх років привернули велику увагу. Огляд задач такого класу представлений в роботі [220], що добре описує проблему та різні її розширення.

В класі задач обробки контейнерів на залізничних сортувальних станціях важливу роль грає задача складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції (scheduling freight trains in rail-rail transshipment yards problem або transshipment yards scheduling problem, TYSP). Ця проблема була вперше сформульована в роботі [221], де описується п’ять рівнів деталізації розв’язання задачі TYSP:

1. Розбиття поїздів на групи і складання графіка їх прибуття на сортувальну станцію (формування сервісних слотів).

2. Призначення поїздів на залізничні колії.

3. Прийняття рішення про розташування контейнерів в поїздах.

4. Прив'язка переміщення контейнерів до портальних кранів (gantry cranes).

5. Прийняття рішення про послідовність переміщень контейнерів портальними кранами.

У статті [221] розв’язується перший рівень проблеми – формування сервісних слотів. В статті наведено математичну модель і два алгоритми розв’язання задачі: точний алгоритм, який використовує динамічне програмування, та евристичний алгоритм, який використовує променевий

пошук (beam search). У [221] також наведено аналіз складності описаних алгоритмів.

Подальші роботи вносять розширення у математичну модель і покращують алгоритми розв’язання. Робота [222] розширює початкову задачу TYSP новими обмеженнями з реального світу і наводить декілька нових алгоритмів розв’язання задачі. Стаття [223] удосконалює алгоритм гілок та меж, розроблений в [222] за допомогою більш ефективної лагранжевої нижньої межі (Lagrangian lower bound). У роботі [223] пропонуються методи отримання двох нових нижніх меж оптимального значення цільової функції. Перша межа отримується на основі релаксованої задачі лінійного програмування, друга – в результаті застосування методу лагранжевої релаксації.

1.7 Постановка задачі дослідження

Незважаючи на велику кількість робіт в області комбінаторної генерації та оптимізації, багато задач залишаються нерозв’язаними. Зокрема, з наведеного вище огляду предметної області можна зробити наступні висновки:

1. Задачі генерації комбінаторних конфігурацій виникають при моделюванні та розв’язанні багатьох задач комбінаторної оптимізації, тому розробка методів та алгоритмів генерації комбінаторних конфігурацій є актуальною проблемою.
2. Опис багатьох класів задач комбінаторної оптимізації вимагає побудови некласичних комбінаторних конфігурацій, серед яких можна виділити *k*-множини, перестановки із заданими підйомами та спусками.
3. Апарат *k*-множин досліджено досить докладно [42]–[44], розглянуті загальні принципи їх генерації [44], однак сама задача генерації *k*-

множин в загальному випадку залишилася нерозв’язаною, досліджений лише один з її окремих випадків [45].

4. Задача генерації *k*-множин, в свою чергу, вимагає розв’язання задачі генерації базових комбінаторних множин. Базовими можуть бути як класичні комбінаторні множини (перестановки, сполучення, розміщення, кортежі), так і некласичні, наприклад, перестановки з заданою кількістю циклів і т.д. [43]–[45]. Для багатьох базових комбінаторних множин описані алгоритми їх генерації [25]–[28], проте в більшості випадків кожний алгоритм генерації ґрунтується на специфічних властивостях конкретної комбінаторної множини, і важко піддається модифікації. Тому виникає необхідність в створенні стратегії та узагальненого методу генерації базових комбінаторних множин широких класів.
5. Незважаючи на значну кількість робіт, присвячених перестановкам з заданими підйомами та спусками [65], [132]–[137], досі нерозв’язаною залишається задача перечислення та генерації перестановок з n елементів, де сигнатуру задано не повністю для всіх n-1 позицій, а частково, тобто лише для деяких з них.
6. Серед задач комбінаторної оптимізації важливим класом є задачі транспортної маршрутизації, зокрема, задачі вивозу та доставки (Pickup and Delivery Problem, PDP). Важливе місце займають задачі вивозу та доставки, що враховують тривимірні обмеження завантаження, можливість блокування одних контейнерів іншими під час розвантаження, крихкість вантажів, стійкість завантаження. Зазвичай математичні моделі задач вивозу та доставки використовують булеві змінні, в той час як опис задачі в термінах комбінаторних конфігурацій відкриває можливість застосовувати численні напрацювання в області комбінаторної генерації та оптимізації для генерації множини допустимих розв’язків та розв’язання таких задачі. Крім того, під час розв’язання таких

витратних з точки зору обчислювальної складності задач, як задачі вивозу та доставки, отримати точний розв’язок в прийнятний час часто буває неможливо, тому використовують різні евристики та метаевристики. Але навіть евристичні алгоритми можуть працювати занадто довго, що не є прийтятним при оперативному плануванні маршрутів; може бути і протилежна ситуація, коли алгоритм може відпрацювати дуже швидко і видати результати, далекі від оптимальних. Таким чином, в реальних умовах буває необхідно регулювати баланс між часом роботи алгоритму та точністю отриманих результатів, але існуючі алгоритми, що розв’язують задачі вивозу та доставки, важко піддаються такому налаштуванню або не піддаються зовсім.

7. Незважаючи на значні результати в області розв’язання задачі складання розкладу руху вантажних поїздів на сортувальних станціях (Transshipment Yards Scheduling Problem, TYSP) [220]–[223], слід зазначити, що [221] і подальші роботи розв’язують тільки перший рівень задачі – формування сервісних слотів, в той час як розв’язання даної задачі на другому рівні, описаному в [221], тобто з урахуванням призначення поїздів на колії, може скоротити витрати на функціонування портальних кранів. До того ж, у існуючих працях задача [221] описується за допомогою булевих змінних, в той час як її опис за допомогою апарату комбінаторних конфігурацій дав би можливість розглядати TYSP як задачу комбінаторної оптимізації і, відповідно, застосовувати добре розроблені методи розв’язання такої задачі. Тому актуальною проблемою є побудова математичної моделі, що використовує математичний апарат комбінаторних конфігурацій, та методу розв’язання задачі складання розкладу руху вантажних поїздів на сортувальних станціях з урахуванням призначення поїздів на колії.

Задача дослідження. На основі зроблених висновків можна сформулювати задачу дослідження: розробити стратегію, методи та алгоритми генерації комбінаторних конфігурацій з метою використання результатів в математичному та комп'ютерному моделюванні для розв'язання задач перевезення та обробки вантажів, що мають комбінаторну природу.

Як математична модель розглянутих в роботі комбінаторних оптимізаційних задач використовується модель вигляду

$$\begin{gathered} k(w) \to \underset{a \in A}{extr}, \\ F(w) \geq 0, \\ G(w) = 0, \\ w \in W,\ W = \{w = (a,d),\ a \in A \subseteq P,\ d \in D\} \end{gathered} \tag{1.1}$$

де W – область допустимих розв'язків задачі, k – відображення, задане на підмножині A деякої комбінаторної множини P, G та F – системи обмежень рівності та нерівності відповідно. Відображення k може залежати від множини додаткових дискретних або неперервних параметрів D, на які можуть бути накладені обмеження для врахування специфіки задач при побудові їх математичних моделей.

Для розв'язання задачі (1.1) в роботі використовується стратегія, що полягає в описі областей допустимих розв'язків за допомогою некласичних комбінаторних конфігурацій та застосуванні комбінаторної генерації в методах розв'язання таких задач.

1.8 Висновки по першому розділу

У розділі проаналізовано існуючі методи комбінаторної генерації та комбінаторної оптимізації. Наведено огляд класичних задач комбінаторики. Проведено аналіз досліджень в галузі комбінаторної генерації, зокрема, методів побудови алгоритмів комбінаторної генерації. Проаналізовано

роботи, присвячені некласичним комбінаторним конфігураціям, таким як композиційні k-образи комбінаторних множин (k-множини) та перестановки з заданими підйомами та спусками. Проведено аналіз публікацій, присвячених задачам та методам комбінаторної оптимізації. Виділено в окремий клас задачі перевезення та обробки вантажів, до яких належать задачі маршрутизації транспорту (наприклад, задача вивозу і доставки) та задачі інтермодальних перевезень, зокрема, задача складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції. Виділено актуальні для предметної області проблеми та сформульовано задачу дослідження дисертаційної роботи, наведено математичну модель розглянутих в роботі задач комбінаторної оптимізації.

2 СТРАТЕГІЇ ГЕНЕРАЦІЇ КОМБІНАТОРНИХ КОНФІГУРАЦІЙ

У даному розділі описуються стратегії, методи та алгоритми генерації деяких некласичних комбінаторних конфігурацій, запропоновано стратегію та узагальнений метод генерації комбінаторних множин.

2.1 Стратегія та узагальнений метод генерації комбінаторних конфігурацій

У даному пункті запропоновано стратегію та узагальнений метод генерації комбінаторних конфігурацій, які є базовим результатом дисертаційної роботи.

На основі аналізу та в розвиток існуючих стратегій комбінаторної генерації запропоновано стратегію генерації комбінаторних конфігурацій, що полягає у:

- використанні єдиного методу генерації комбінаторних конфігурацій;
- визначенні порядку генерації (наприклад, лексикографічний, антилексикографічний і т.д.);
- визначенні повноти генерації комбінаторної множини (вичерпна або часткова генерація).

В рамках запропонованої стратегії генерації розроблено узагальнений метод генерації комбінаторних конфігурацій, що ґрунтується на класичній процедурі backtracking [28], яка полягає в утворенні від часткового кортежу довжини i одного або декількох кортежів довжини i+1. При цьому порядок та повнота генерації задаються правилами побудови конкретної комбінаторної множини. Опишемо цей метод.

З кожною базовою комбінаторною множиною T пов`яжемо множину p, що містить правила побудови конкретної комбінаторної множини (які описуються нижче), а також специфічні для цієї множини параметри. Для

найпоширеніших комбінаторних множин (перестановки, сполучення, розміщення) *p* містить такі параметри:

- множину твірних елементів $A=\{a_1,a_2,...,a_n\}$, $a_1<a_2<...<a_n$;
- m – довжину кожного елементу множини, тобто кортежу $t\in T$.

Для некласичних комбінаторних множин *p* може містити інші параметри, наприклад, сигнатуру *Q* для перестановок с заданими підйомами та спусками [134], [137] або параметри $n_1,n_2,...,n_k$ для перестановок з повтореннями та інші параметри, що є характерними для різних класів комбінаторних множин. Під класом комбінаторної множини будемо розуміти її належність до перестановок, сполучень і т.д.

Нехай задана базова комбінаторна множина T та її параметри *p*. Необхідно згенерувати всі елементи $t\in T$, кожний з яких є кортежем довжини *m*. Позначимо $t^i=(t_1,t_2,...,t_i)$, $\forall t_i\in A$, $A\in p$, $i\in J_n$. Тоді $t^0=(\ )$, $t^m=t\in T$. Результатом генерації є множина T.

Опишемо принцип роботи алгоритму *GenBase,* що реалізує узагальнений метод комбінаторної генерації. Алгоритм працює рекурсивно, на кожному рівні рекурсії $i\in J^0_{m-1}$, $J^0_{m-1}=\{0,1,...,m-1\}$ додаючи в поточний елемент-кортеж $t^i=(t_1,t_2,...,t_i)$ наступний твірний елемент $t_{i+1}\in A$, отримуючи таким чином на рівні $i+1$ кортеж $t^{i+1}=(t_1,t_2,...,t_{i+1})$. На рівні *m* алгоритм додає повний кортеж $t^m=t$ до результуючої множини *T*. Таким чином, у процесі своєї роботи алгоритм будує рекурсивне дерево (backtracking tree) [28], [118].

Належність множини T до конкретного класу комбінаторних множин задає відповідні обмеження на елемент $t_{i+1}\in A$. На кожному рівні $i\in J^0_{m-1}$ позначимо через $F^i=(f_1,f_2,...,f_k)$, $\forall f_j\in A$, $j=1,2,...,k$ кортеж, що містить всі твірні елементи, що задовольняють цим обмеженням.

Правила побудови конкретної комбінаторної множини, тобто правила отримання F^i, задаються за допомогою процедури *Get_F* (що входить до

множини *p*). Після отримання F^i за допомогою процедури *Get_F* , алгоритм додає у кінець вхідного кортежу $t^i = (t_1, t_2, ..., t_i)$ елемент $t_{i+1} = f_j$ і рекурсивно викликає сам себе з параметром $t^{i+1} = (t_1, t_2, ..., f_j)$ для всіх $j \in J_k$.

Узагальненість розробленого методу генерації полягає саме в процедурі *GetF*, специфічній для кожної комбінаторної множини. Якщо для довільної комбінаторної множини сконструювати таку функцію, то її можна генерувати за допомогою запропонованого методу.

Алгоритм *GenBase* використовує ту саму ідею побудови рекурсивного дерева, що і алгоритм пошуку з поверненням (backtracking search), описаний Е. Рейнгольдом в [118], але застосовує її для генерації комбінаторних множин, а не для пошуку.

Вхідні дані алгоритму – множина T (до якої будуть додаватися згенеровані елементи), поточний частковий кортеж t^i, а також набір параметрів p.

Для генерації всіх елементів множини *T*, алгоритм викликається з параметрами $T = (\,)$, $t^0 = (\,)$, p.

def *GenBase*(T, t^i, p)
  if $i == m$
    $T = T \cup t^i$
    return
  F^i = *Get_F*(T, t^i, p)
  for $j = 1, 2, ..., |F^i|$
    GenBase(T, $t^{i+1} = (t_1, t_2, ..., t_i, f_j)$, p)

Як вже зазначалося, процедура *GetF* задається окремо для кожного класу комбінаторних множин і передається у множині параметрів p. Ця функція також визначає порядок та повноту генерації комбінаторної множини.

Сконструюємо процедуру *GetF* для деяких поширених класів комбінаторних множин.

Для розміщень з повтореннями кортеж F^i містить усі твірні елементи

$$F^i = A.$$

Відповідна процедура *GetF_arrangements_with_repetitions* наведена нижче.

def *GetF_arrangements_with_repetitions*(T, t^i, p)

return A

Для розміщень без повторень і перестановок як їх частного випадку до кортежу F^i входять $n-i$ твірних елементів, що не входять в поточний кортеж t^i

$$F^i = (f_1, f_2, \ldots f_l, \ldots, f_{n-i}):\ f_l \neq t_j, \forall j \in J_i, \forall l \in J_{n-i}, J_i = \{1,2,\ldots,i\}, J_{n-i} = \{1,2,\ldots,n-i\}.$$

Відповідна процедура *GetF_arrangements* наведена нижче.

def *GetF_arrangements* (T, t^i, p)

return $(f_1, f_2, \ldots, f_{n-i}): f_l \neq t_j, \forall j \in J_i, \forall l \in J_{n-i}$

В сполученнях порядок елементів не є важливим, тому їх можна генерувати у вигляді впорядкованих кортежів $t^i = (t_1, t_2, \ldots, t_i)$, де $t_1 < t_2 < \ldots < t_i$ для сполучень без повторень і $t_1 \leq t_2 \leq \ldots \leq t_i$ для сполучень з повтореннями.

Тоді для сполучень без повторень F^i містить усі твірні елементи, що не входять до t^i і є більшими за t_i:

$$F^i = (f_1, f_2, \ldots, f_k):\ f_l \in A,\ f_l \neq t_j,\ f_l > t_i, \forall l \in J_k, \forall j \in J_{i-1}.$$

У випадку сполучень з повтореннями до F^i також входять твірні елементи, що дорівнюють t_i

$$F^i = (f_1, f_2, ..., f_k): f_l \in A, f_l \neq t_j, f_l \geq t_i, \forall l \in J_k, \forall j \in J_{i-1}.$$

Відповідні процедури *GetF_combinations* та *GetF_combinations_with_repetitions* наведено нижче.

def *GetF_combinations* (T, t^i, p)

return $(f_1, f_2, ..., f_k): f_l \in A, f_l \neq t_j, f_l > t_i, \forall l \in J_k, \forall j \in J_{i-1}$

def *GetF_combinations_with_repetitions* (T, t^i, p)

return $(f_1, f_2, ..., f_k): f_l \in A, f_l \neq t_j, f_l \geq t_i, \forall l \in J_k, \forall j \in J_{i-1}$

Описаний узагальнений метод генерації є базовим результатом даної роботи, на якому засновані всі подальші представлені у роботі методи комбінаторної генерації та оптимізації. Також метод використовувався у роботі [59] для побудови евристичного методу розв'язання задачі оптимізації лінійної функції на множині циклічних перестановок.

Оцінка складності узагальненого алгоритму генерації. Виходячи з методів оцінки складності рекурсивного алгоритму, слідуючи Ф. Раскі [27], під складністю алгоритму будемо розуміти кількість змін проміжних даних, що відбулися з моменту виклику алгоритму до моменту закінчення його роботи.

В алгоритмі *GenBase* до вхідного часткового кортежу t^i на кожному рівні рекурсії $i \in J_{m-1}^0$ додаються елементи множини F^i і алгоритм *GenBase* викликає себе рекурсивно, тобто відбувається змінення проміжних даних (кортежів t^i), і тому складність може біти оцінена як кількість рекурсивних викликів *GenBase* на рівнях $i \in J_{m-1}^0$.

На кожному рівні рекурсії в кортеж t^i додається один елемент, тому для генерації кожного з N кортежів $t^m = t \in T$ *GenBase* викликається не більш ніж

m раз. Тоді складність генерації елементів однієї базової множини не перевищує mN, $N = Card(T)$.

Таким чином, складність узагальненого алгоритму генерації визначається як

$$O(N) = mN, \qquad \textbf{(}2.1\textbf{)}$$

де N – потужність множини T,

m – довжина кожного елемента-кортежа множини T.

На основі оцінки складності узагальненого алгоритму *GenBase*, можна зробити висновок, що він, за класифікацією алгоритмів комбінаторної генерації, запропонованій F. Ruskey [27], належить до класу BEST (Backtracking Ensuring Success at Terminals).

2.2 Генерація композиційних *k*-образів комбінаторних множин

В даному пункті розглядається загальний підхід до генерації композиційних $k-$образів комбінаторних множин ($k-$множин) на основі використання описаних вище узагальнених стратегії та методу для генерації базових комбінаторних множин.

Розроблено метод та алгоритм *Gen_k-set* генерації *k*-множин, що використовують узагальнений метод та алгоритм *GenBase* для генерації базових множин. Оцінено обчислювальну складність розробленого алгоритму.

2.2.1 Математичний опис k-множин

Нехай задано множини довільних елементів

$$z^{\beta} = \{z_1^{\beta}, z_2^{\beta}, ..., z_{n_\beta}^{\beta}\} \subseteq \mathbf{Z}_{\boldsymbol{\beta}_i},\ \mathbf{Z}_{\boldsymbol{\beta}_i},\ \beta \in \boldsymbol{\beta}_i,\ i \in J_k^0 = \{0,1,...,k\},$$

де

$$\boldsymbol{\beta}_0 = \{0\},\ \boldsymbol{\beta}_i = \{\beta_j,\ j = 1,2...,\eta_i\},\ \beta_j = (\alpha_1, \alpha_2, ..., \alpha_i),$$

$$\alpha_1 \in J_n,\ \alpha_2 \in J_{n_{\alpha_1}}, ..., \alpha_i \in J_{n_{\alpha_1 ... \alpha_{i-1}}}$$

та

$$\eta_1 = n,\ \eta_2 = \sum_{j=1}^{n} n_j,\ \eta_i = \sum_{\alpha_1=1}^{n} \sum_{\alpha_2=1}^{n_1} ... \sum_{\alpha_{i-1}=1}^{n_{1...i-2}} n_{\alpha_1 ... \alpha_{i-1}},\ i = 3,4,...,k. \qquad \textbf{(2.2)}$$

Розглянемо відображення [43], [44]

$$\Gamma_{\boldsymbol{\beta}_0}: \mathbf{Z}_{\boldsymbol{\beta}_0} \to \mathbf{Y}^0,\ \Gamma_{\boldsymbol{\beta}_i}: \mathbf{Y}^{i-1} \times \mathbf{Z}_{\boldsymbol{\beta}_i} \to \mathbf{Y}^i,$$

де

$$\mathbf{Y}^0 = \{Y_{\boldsymbol{\beta}_0}(z^\beta), \beta \in \boldsymbol{\beta}_0\}, \mathbf{Y}^i = \{Y_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}}, z^\beta), \beta \in \boldsymbol{\beta}_i\},\ i \in J_k,\ J_t = \{1,2,...,t\},$$

$$Y_{\boldsymbol{\beta}_i} = \Gamma_{\boldsymbol{\beta}_i}(Y_{\boldsymbol{\beta}_{i-1}}, \mathrm{z}^{\boldsymbol{\beta}_i}) = F(Y_{\boldsymbol{\beta}_{i-1}}, \tilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i})),\ i \in J_k,\ F(Y_{\boldsymbol{\beta}_{i-1}}, \tilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}))$$

є відображеннями, що реалізують операцію n-заміщення, що полягає в заміні кожного твірного елементу множини $Y_{\boldsymbol{\beta}_{i-1}}$ елементами базових комбінаторних множин $Y_\beta = \tilde{\Gamma}_\beta(\mathrm{z}^\beta),\ \beta \in \boldsymbol{\beta}_i$, відповідно,

$$\tilde{\Gamma}_{\boldsymbol{\beta}_i}(\mathrm{z}^{\boldsymbol{\beta}_i}) = (\tilde{\Gamma}_\beta(\mathrm{z}^\beta),\ \beta \in \boldsymbol{\beta}_i),\ \mathrm{z}^{\boldsymbol{\beta}_i} = (\mathrm{z}^\beta,\ \beta \in \boldsymbol{\beta}_i),\ \tilde{\Gamma}_\beta(\mathrm{z}^\beta)$$

є базовими відображеннями [43], [44]. Це означає, що

$$(z_{l_1}^\beta, z_{l_2}^\beta, ..., z_{l_\beta}^\beta) \in Y_\beta,\ z_{l_t}^\beta \in Y_\delta,\ l_t \in J_{l_\beta},\ \beta \in \boldsymbol{\beta}_{i-1},\ \delta \in \boldsymbol{\beta}_i,\ i \in J_k.$$

Позначимо

$$\boldsymbol{\Gamma}_i = \{\Gamma_{\boldsymbol{\beta}_i}\},\ i \in J_k. \qquad (2.3)$$

Визначення. Композиційним $k-$образом комбінаторних множин Y_0, Y_1, $Y_2,\ldots,Y_n$, $Y_{11},Y_{12},\ldots,Y_{1n_1}$, …, $\underset{k}{Y_{1\ldots1}},\ldots,Y_{nn_1\ldots n_{k-1}}$ ($k-$множиною), утвореним множинами z^{β_k}, $\beta_k \in \boldsymbol{\beta}_k$, називається комбінаторна множина виду [43], [44]

$$W_z = \Gamma_k \circ \Gamma_{k-1} \circ \ldots \circ \Gamma_0(z), \tag{2.4}$$

де відображення $\Gamma_i \in \boldsymbol{\Gamma}_i$, $i \in J_k$ визначаються відношенням (2.3).

Потужність множини (2.4) визначається відношенням [43], [44]

$$Card(W_z) = \sum_{\{\gamma_1,\gamma_2,\ldots,\gamma_r\}\subset J_n} \alpha_{\gamma_1,\gamma_2,\ldots,\gamma_r} \cdot \prod_{i=1}^{k} \prod_{\beta_i=(\alpha_1\alpha_2\ldots\alpha_i)} Card\ Y_{\beta_i} \tag{2.5}$$

Генерація $k-$множин базується на генерації базових комбінаторних множин, тому необхідно спочатку визначити алгоритм для генерації цих множин. Як було зазначено в пункті 1.7, для генерації базових множин доцільно створити єдиний алгоритм, який дозволяв би отримати комбінаторну множину будь-якого виду.

2.2.2 Генерація *k*-множин

Нехай задані комбінаторні множини Y_0, $Y_1,Y_2,\ldots,Y_n$, $Y_{11},Y_{12},\ldots,Y_{1n_1},\ldots$, $\underset{k}{Y_{1\ldots1}},\ldots,Y_{nn_1\ldots n_{k-1}}$, а також параметри $p(Y_0)$, $p(Y_1),\ldots,p(Y_{nn_1\ldots n_{k-1}})$. Необхідно згенерувати $k-$множину $W_z = \Gamma_k \circ \Gamma_{k-1} \circ \ldots \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$. Введемо поперше деякі позначення. Множину $Y_{i_1 i_2 \ldots i_n}$ будемо називати батьківською множиною по відношенню до множини $Y_{j_1 j_2 \ldots j_n j_{n+1}}$, якщо $i_1 = j_1, i_2 = j_2, \ldots, i_n = j_n$. Множину $Y_{j_1 j_2 \ldots j_n j_{n+1}}$ будемо називати дочірньою множиною множини $Y_{i_1 i_2 \ldots i_n}$. Для зручності подальших викладок, змінимо систему індексації базових

множин, перейшовши від індексів змінної довжини до індексів фіксованої довжини. Кожну базову комбінаторну множину на рівні $i \in J_{k-1}^{0}$ позначимо як Y_{ij}, де $j \in J_{\eta_i}$ – порядковий номер базової множини, а η_i визначається формулою **(**2.2**)**. З міркувань зручності, в подальших викладках будемо позначати Y_0 як Y_{01}.

Приклад 2.1. Нехай задано базові комбінаторні множини Y_0, Y_1, Y_2, Y_{11}, Y_{21}, Y_{22}, $Y_{111}, Y_{211}, Y_{221}$, зв'язок між якими можна схематично зобразити у вигляді дерева на рис. 2.1.

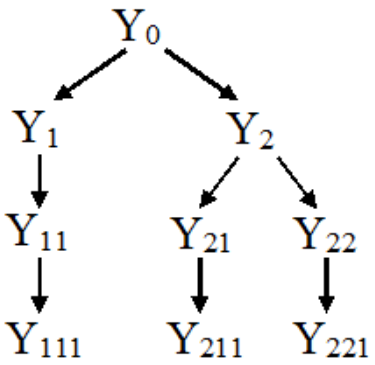


Рисунок 2.1. Схема зв'язку базових множин

На першому рівні такого дерева знаходяться множини Y_1 та Y_2, на другому – Y_{11}, Y_{21} та Y_{22}, на третьому – Y_{111}, Y_{211} та Y_{221}. У відповідності з новою системою індексації, позначимо множини першого рівня як Y_{11} та Y_{12}, другого – Y_{21}, Y_{22} та Y_{23}, третього – Y_{31}, Y_{32} та Y_{33} і т.д.. Тоді схема на рис. 2.1 приймає вигляд з рис. 2.2.

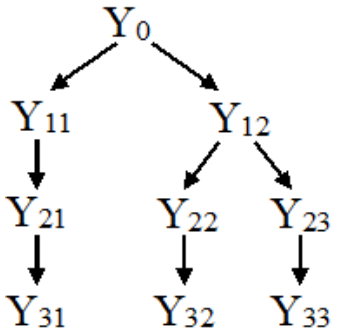


Рисунок 2.2. Схема зв'язку базових множин при зміненій індексації

Запропонований спосіб індексації дозволяє неявно задати зв'язок між дочірньою та батьківською множинами. Нехай множина Y_{ij} утворена елементами множини $A_{ij} \in p(Y_{ij})$, потужність якої $n_{ij} = |A_{ij}|$. Тоді із принципу побудови $k-$множин випливає, що на рівні $i+1$ знаходиться n_{ij} дочірніх множин батьківської множини Y_{ij}. Нехай перші n_{i1} множини на рівні $i+1$ є дочірніми по відношенню до множини Y_{i1}, наступні n_{i2} множин – дочірніми по відношенню до множини Y_{i2} і т.д. для всіх Y_{ij}, $j \in J_{\eta_i}$. Тоді для кожної базової множини можна однозначно встановити всі її батьківські та дочірні множини. При виконанні операції n-заміщення твірний елемент $a_l \in A_{ij}$ батьківської множини Y_{ij} буде заміщений елементами l-ої її дочірньої множини.

Приклад 2.2. Нехай множина представлена деревом, наведеним на рис. 2.3.

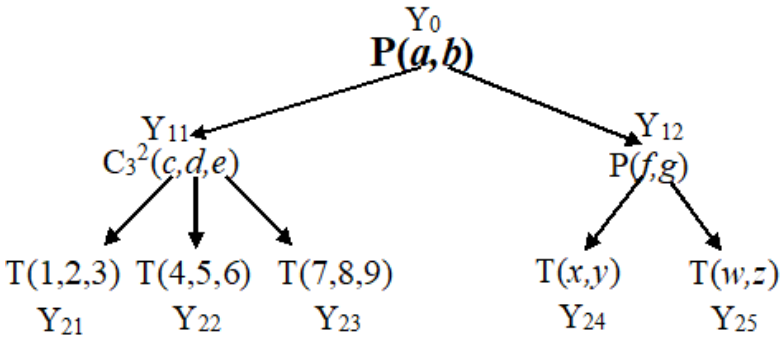


Рисунок 2.3. *k*-множина для прикладу 2.2

Тут $P(a,b)$ – множина перестановок елементів a та b, $C_3^2(c,d,e)$ – множина сполучень з 3 по 2, $T(1,2,3)$ – кортеж і т.д.

Множину породжено трьома елементами c,d,e, які при виконанні операції n-заміщення будуть замінені елементами множин Y_{21}, Y_{22} та Y_{23} відповідно. Утворюючі елементи f та g множини Y_{12} будуть замінені елементами множин Y_{24} та Y_{25} відповідно.

2.2.3 Алгоритм генерації *k*-множин

Опишемо алгоритм генерації $k-$множин. На початку його роботи, за допомогою алгоритму *GenBase* генеруються елементи кожної базової комбінаторної множини Y_{ij}, $i \in J_{k-1}^0$, $j \in J_{\eta_i}$. Після цього послідовно будуються відображення $\Gamma_{i+1} \circ \Gamma_i \circ .. \circ \Gamma_0(z)$, $z = A_0 \in p(Y_0)$ для кожного $i \in J_{k-1}^0$, тобто проводиться операція n-заміщення, де твірні елементи батьківської множини замінюються елементами-кортежами його дочірніх множин.

На рівні $i = 0$ батьківською множиною є множина Y_0, дочірніми множинами – множини $Y_{11}, Y_{12}, ..., Y_{1\eta_1}$. На рівні $i = 1$ батьківська множина – результат композиції відображень $\Gamma_1 \circ \Gamma_0(z)$, отриманий на нульовому рівні, дочірніми множинами – множини $Y_{21}, Y_{22}, ..., Y_{2\eta_2}$. На рівні i батьківська множина – це результат композиції відображень $\Gamma_i \circ \Gamma_{i-1} \circ ... \circ \Gamma_0(z)$, а дочірніми множинами є $Y_{(i+1)1}, Y_{(i+1)2}, .., Y_{(i+1)\eta_{i+1}}$.

Введемо множину P^i, що являє собою результат застосування композиції відображень $\Gamma_i \circ \Gamma_{i-1} \circ ... \circ \Gamma_0$ до вихідної множини $z = A_0 \in p(Y_0)$. Позначимо множину $P^i = \Gamma_i \circ \Gamma_{i-1} \circ ... \circ \Gamma_0(z)$, множину її твірних елементів – A^i, довжину кожного її елемента-кортежа – через m^i. Так як на рівні i множина P^i містить елементи-кортежі множин $Y_{i1}, ..., Y_{i\eta_i}$, то її твірна множина

$$A^i = \bigcup_{j=1}^{\eta_i} A_{ij}, \; A_{ij} \in p(Y_{ij}), \tag{2.6}$$

а довжина кожного елемента-кортежа

$$m^i = \bigcup_{j=1}^{\eta_i} m_{ij}, m_{ij} \in p(Y_{ij}), \tag{2.7}$$

де A_{ij} та m_{ij} – відповідно множина твірних елементів та довжина елемента-кортежа базової множини Y_{ij}.

Для нульового рівня $P^0 = Y_0$, $A^0 = A_0 \in p(Y_0)$, $m^0 = m_0 \in p(Y_0)$.

Для обчислення η_i при новій індексації можна застосовувати формулу

$$\eta_i = \sum_{j=1}^{\eta_{i-1}} |\, A_{(i-1)j} \,|, \quad A_{(i-1)j} \in p(Y_{(i-1)j}), \tag{2.8}$$

аналогічну формулі (2.2). В операції n-заміщення елементарною операцією є заміна одного кортежу іншим. Опишемо функцію ***replace*,** що виконує таку заміну. Входом функції є кортеж $x = (x_1, x_2, ..., x_m)$, його довжина m, множина елементів $A = \{a_i\}$, якими утворений кортеж x, та множина $R = \{r_i\}, i \in J_n, \; J_n = \{1, 2, ..., n\}, n = |\, A \,|$, така, що кожний елемент $a_i \in A$ необхідно замінити на $r_i \in R$. Функція повертає кортеж x, в якому виконано всі необхідні заміни.

def *replace*(x, m, A, R):
for $i = 1, 2, ..., |\, A \,|$
    for $j = 1, 2, ..., m$
        if $x_j == a_i$ **then** $x_j = r_i$
return x

Вхідними даними алгоритму генерації *k*-множин *Gen_k-set* є число k (число рівнів дерева відповідно дорівнює $k+1$), класи базових множин Y_{ij} і їхні параметри $p(Y_{ij})$, $i \in J_k^0$, $j \in J_{\eta_i}$. Результатом роботи алгоритму є послідовність елементів згенерованої $k-$множини.

Алгоритм *Gen_k-set* генерації *k*-множин наведено нижче.

def *Gen_k-set*$(k, \{Y_{ij}\}, \{p(Y_{ij})\})$

$\eta_0 = 1$

for $i = 0,1,...,k$

if $i \neq 0$

$$\eta_i = \sum_{j=1}^{\eta_{i-1}} |A_{(i-1)j}|$$

for $j = 1,2,...,\eta_i$

$$Y_{ij} = Gen\_Base(Y_{ij}, (\,), p(Y_{ij}))$$

$P^0 = Y_0$

$A^0 = A \in p(Y_0)$

$m^0 = m \in p(Y_0)$

for $i = 0,1,...,k-1$

$P^{i+1} = \varnothing$

$$A^i = \bigcup_{j=1}^{\eta_i} A_{ij}$$

$$m^i = \bigcup_{j=1}^{\eta_i} m_{ij}$$

for $x \in P^i$

for $s_1 \in Y_{(i+1)1}$

for $s_2 \in Y_{(i+1)2}$

.....

for $s_{\eta_{i+1}} \in Y_{(i+1)\eta_{i+1}}$

$P^{i+1} = P^{i+1} \cup replace(x, m^i, A^i, \{s_1, s_2, ..., s_{\eta_{i+1}}\})$

if $i + 1 == k$

return P^{i+1}

На кожному рівні $i \in J_{k-1}^0$ в кожному елементі-кортежі поточної батьківської множини P^i твірні елементи замінюються всіма можливими комбінаціями елементів-кортежів дочірніх множин $Y_{(i+1)1}, Y_{(i+1)2}, .., Y_{(i+1)\eta_{i+1}}$. Проходячи в циклі по всім елементам-кортежам кожної дочірньої множини, кожен раз отримуємо набір кортежів $\{s_1, s_2, ..., s_{\eta_{i+1}}\}$, де $s_j \in Y_{(i+1)j}$ – поточний

кортеж *j*-го дочірньої множини, і за допомогою функції *replace* в кортежі $x \in P^i$ замінюємо кожен твірний елемент $a_j \in A^i$ на $s_j \in Y_{(i+1)j}$.

Істотною перевагою алгоритму є можливість отримання проміжних результатів, тобто множин P^i на кожному рівні $i \in J_{k-1}$, які можна розглядати як $k-$множини, де $k = i$.

Приклад 2.3. Нехай необхідно згенерувати $k-$множину, структура якої представлена на рис. 2.4. Дана $k-$множина містить два рівні ієрархії і є композицією перестановок ($k = 1$). Множина Y_0 – це множина перестановок двох елементів або розміщень з двох елементів по два, а $m_0 = 2$. Утворюючі елементи Y_0 можуть бути будь-якими, тому $A_0 = \{a,b\}$. Тоді $Y_0 = \{(a,b),(b,a)\}$. На нижньому рівні $A_{11} = \{1,2\}, A_{12} = \{3,4\}$, $m_{11} = m_{12} = 2$, $Y_{11} = \{(12)(21)\}$, $Y_{12} = \{(34)(43)\}$.

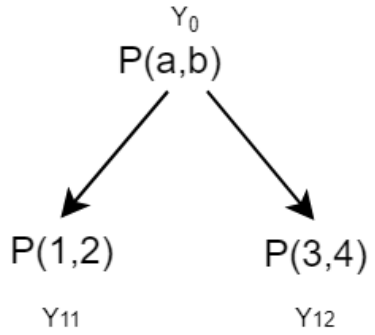


Рисунок 2.4. Структура $k-$множини для прикладу 2.3

В операції n-заміщення в батьківській множині Y_0 перший твірний елемент a буде замінений елементами множини Y_{11}, тобто (12) та (21), а елемент b – елементами (34) та (43).

Розглянемо хід отримання множини $W_z = \Gamma_1 \circ \Gamma_0(z) = P^1$:

1 $x = (ab)$

1.1 $s_1 = (12)$

1.1.1 $s_2 = (34)$

$P^1 = \varnothing \cup replace((ab),2,\{a,b\},\{(12),(34)\})$

$\# P^1 = \{(1234)\}$

1.1.2 $s_2 = (43)$

$P^1 = \varnothing \cup replace((ab),2,\{a,b\},\{(12),(43)\})$

$\#P^1 = \{(1234),(1243)\}$

1.2 $s_1 = (21)$

1.2.1 $s_2 = (34)$

$P^1 = \varnothing \cup replace((ab),2,\{a,b\},\{(21),(34)\})$

$\#P^1 = \{(1234),(1243),(2134)\}$

1.2.2 $s_2 = (43)$

$P^1 = \varnothing \cup replace((ab),2,\{a,b\},\{(21),(43)\})$

$\#P^1 = \{(1234),(1243),(2134),(2143)\}$

2 $x = (ba)$

2.1 $s_1 = (12)$

2.1.1 $s_2 = (34)$

$P^1 = \varnothing \cup replace((ba),2,\{a,b\},\{(12),(34)\})$

$\#P^1 = \{(1234),(1243),(2134),(2143),$
$(3412)\}$

2.1.2 $s_2 = (43)$

$P^1 = \varnothing \cup replace((ba),2,\{a,b\},\{(12),(43)\})$

$\#P^1 = \{(1234),(1243),(2134),(2143),$
$(3412),(4312)\}$

2.2 $s_1 = (21)$

2.2.1 $s_2 = (34)$

$P^1 = \varnothing \cup replace((ba),2,\{a,b\},\{(21),(34)\})$

$\#P^1 = \{(1234),(1243),(2134),(2143),$
$(3412),(4312),(3421)\}$

2.2.2 $s_2 = (43)$

$P^1 = \varnothing \cup replace((ba),2,\{a,b\},\{(21),(43)\})$

$\#P^1 = \{(1234),(1243),(2134),(2143),$
$(3412),(4312),(3421),(4321)\}$

В результаті отримаємо

$$W_z = \Gamma_1 \circ \Gamma_0(z) = P^1 = \{(1234),(1243),(2134),(2143),$$
$$(3412),(4312),(3421),(4321)\}.$$

2.2.4 Оцінка складності алгоритму Gen_k-set

Складність алгоритму *Gen_k-set* визначається складністю генерації базових множин, а також складністю виконання операцій *n*-заміщення та кількістю рівнів $k-$множини.

Для генерації базових множин можуть бути застосовані як відомі алгоритми з відомими оцінками складності (наприклад, [25], [26], [28], [110]), так і описаний в даній роботі узагальнений алгоритм *GenBase*. В будь-якому випадку кожна базова комбінаторна множина Y_{ij}, $i \in J_k^0$, $j \in J_{\eta_i}$ повинна бути знегерована одним з алгоритмів. Тоді складність генерації базових множин визначається як

$$\sum_{i=0}^{k}\sum_{j=1}^{\eta_i} O(Y_{ij}), \tag{2.9}$$

де $O(Y_{ij})$ – складність генерації базової множини Y_{ij}, що залежить від використаного алгоритму.

Якщо для генерації всіх базових множин використовувати алгоритм *GenBase* (оцінка складності якого наведена в формулі **(**2.1**)**), то формула (2.9) приймає вигляд

$$\sum_{i=0}^{k}\sum_{j=1}^{\eta_i} m_{ij} Card(Y_{ij}), \quad m_{ij} \in p(Y_{ij}). \tag{2.10}$$

Слідуючи обраному способу оцінки складності, будемо визначати складність виконання операцій *n*-заміщення в алгоритмі *Gen_k-set* кількістю викликів функції *replace*, що змінює проміжні дані алгоритму, а саме кортежи $x \in P^i$. Із принципу роботи алгоритму *Gen_k-set* слідує, що на кожному рівні $k-$множини $i \in J_0^{k-1}$ функція *replace* викликається

$$M = Card(P^i) \cdot Card(Y_{(i+1)1}) \cdot Card(Y_{(i+1)2}) \cdot \ldots \cdot Card(Y_{(i+1)\eta_{i+1}}) \qquad (2.11)$$

раз. Величина $Card(P^i)$ може бути отримана із формули (2.5) при підстановці $k = i$. Величини $Card\, Y_{(i+1)j}$, $j \in J_{\eta_{i+1}}$ – це потужності базових множин, що визначаються за допомогою відомих для кожного класу комбінаторних множин формул.

На всіх рівнях $i \in J_0^{k-1}$ кількість викликів функції *replace* дорівнює

$$\sum_{i=0}^{k-1} (Card(P^i) \prod_{j=1}^{\eta_{i+1}} Card(Y_{(i+1)j})). \qquad (2.12)$$

Таким чином, справедливим є наступне твердження.

Лема 2.1. Обчислювальна складність алгоритму генерації $k-$множин *Gen_k-set* дорівнює

$$\sum_{i=0}^{k} \sum_{j=1}^{\eta_i} O(Y_{ij}) + \sum_{i=0}^{k-1} (Card(P^i) \prod_{j=1}^{\eta_{i+1}} Card(Y_{(i+1)j})). \qquad (2.13)$$

Для випадку, визначеного в [44] як $k-$композиція перестановок, коли всі базові множини є перестановками, може бути отримана спрощена формула для безпосереднього перечислення кількості елементів в $k-$множині. Із результатів [44] можна отримати формулу для $Card(P^i)$. Якщо всі множини Y_{ij} є перестановками з n_{ij} елементів, тоді $Card(Y_{ij}) = n_{ij}!$ та

$$Card(P^i) = \prod_{u=0}^{i} \prod_{j=1}^{\eta_i} (n_{uj})! \qquad (2.14)$$

Тоді формула (2.11) приймає вигляд

$$\sum_{i=0}^{k-1}\left(\prod_{u=0}^{i}\prod_{j=1}^{\eta_i}(n_{uj})!\cdot\prod_{v=1}^{\eta_{i+1}}(n_{(i+1)v})!\right). \qquad (2.15)$$

Обчислювальна складність алгоритму *Gen_k-set* у цьому випадку дорівнює

$$\sum_{i=0}^{k}\sum_{j=1}^{\eta_i}O(Y_{ij})+\sum_{i=0}^{k-1}\left(\prod_{u=0}^{i}\prod_{j=1}^{\eta_i}(n_{uj})!\cdot\prod_{v=1}^{\eta_{i+1}}(n_{(i+1)v})!\right), \qquad (2.16)$$

де n_{ij} – кількість твірних елементів множини Y_{ij}.

2.3 Випадкова генерація k-множин

Як було зазначено в пункті 2.2, за допомогою апарату *k*-множин можуть бути описані множини допустимих розв'язків деяких задач, що мають композиційну структуру (наприклад, [39]–[41]). Відповідно, метод генерації *k*-множин може бути використаний для генерації множини допустимих розв'язків таких задач. Проте для реальних задач, що мають велику розмірність, генерація усіх елементів *k*-множини може займати неприйнятно великий час. Тому виникає необхідність у генерації випадковим чином сформованої підмножини з усіх елементів *k*-множини. Одним з методів генерації такої вибірки може служити генерація випадкових елементів *k*-множин, або випадкова генерація *k*-множин.

Даний пункт присвячений опису методу та алгоритму випадкової генерації *k*-множин.

2.3.1 Метод випадкової генерації k-множин

Описаний в підпункті 2.2.2 метод генерації *k*-множин, реалізований за допомогою алгоритму *Gen_k-set,* в операції *n*-заміщення виконує послідовну

заміну елемента батьківської множини на кожен елемент дочірньої множини. Метод випадкової генерації *k*-множин полягає в тому, що операції *n*-заміщення виконуються не для всіх, а лише для деяких елементів дочірньої множини. Для кожної базової множини Y на рівні $i \in J_k$, до множини параметрів p додамо величину $0 \le Q \le 1$, що описує відносну долю кількості елементів дочірьої множини, що приймають участь в операціях *n*-заміщення. Кількість дочірніх елементів обчислюється як

$$V(Y) = |Y| \cdot Q,$$

де $|Y|$ – потужність дочірньої множини.

Алгоритм *Gen_Random_k-set* для випадкової генерації *k*-множин представлено нижче. Цей алгоритм є модифікацією алгоритму повної генерації *k*-множин *Gen_k-set*, описаного в підпункті 2.2.3. Випадковий вибір декількох елементів з дочірніх множин виконується функцією *rand_sample*, реалізація якої є тривіальною задачею. Наприклад, у мові Python ця функція реалізована в стандартному модулі *random* і доступна як *random.sample* [224].

def *Gen_Random_k-set*$(k, \{Y_{ij}\}, \{p(Y_{ij})\})$
$\eta_0 = 1$
for $i = 0,1,...,k$
 if $i \neq 0$
  $\eta_i = \sum_{j=1}^{\eta_{i-1}} |A_{(i-1)j}|$
 for $j = 1,2,...,\eta_i$
  Y_{ij} = *Gen_Base*$(Y_{ij}, (\ ), p(Y_{ij}))$
$P^0 = Y_0$
$A^0 = A \in p(Y_0)$
$m^0 = m \in p(Y_0)$
for $i = 0,1,...,k-1$ **do**
 $P^{i+1} = \varnothing$

$$A^i = \bigcup_{j=1}^{\eta_i} A_{ij}$$

$$m^i = \bigcup_{j=1}^{\eta_i} m_{ij}$$

for $x \in P^i$

  for $s_1 \in rand\_sample(Y_{(i+1)1}, Q \cdot | Y_{(i+1)1} |)$

    for $s_2 \in rand\_sample(Y_{(i+1)2}, Q \cdot | Y_{(i+1)2} |)$

      .....

        for $s_{\eta_{i+1}} \in rand\_sample(Y_{(i+1)\eta_{i+1}}, Q \cdot | Y_{(i+1)\eta_{i+1}} |)$

          $P^{i+1} = P^{i+1} \cup replace(x, m^i, A^i, \{s_1, s_2, ..., s_{\eta_{i+1}}\})$

if $i+1 == k$

  return P^{i+1}

Приклад 2.4. Опишемо процес роботи алгоритму *Gen_Random_k-set* на прикладі генерації композиції перестановок – *k*-множини, структура якої представлена на рис. 2.5.

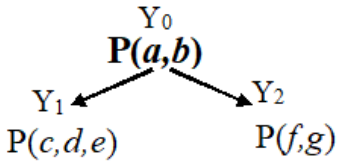


Рисунок 2.5. Представлення k – множини для прикладу 2.4

Утворюючі елементи *a* та *b* множини $Y_0 = \{(ab),(ba)\}$ замінюються твірними елементами дочірніх множин $Y_1 = \{(cde),(ced),(dce),$ $(dec),(ecd),(edc)\}$ та $Y_2 = \{(fg),(gf)\}$ відповідно. Нехай $Q(Y_1) = Q(Y_2) = 0.5$. Звідси, $V(Y_1) = 3;\ V(Y_2) = 1$, тобто твірний елемент *a* батьківської множини Y_0 послідовно замінюється на кожен з трьох елементів дочірньої множини Y_1, обраних випадковим чином, а твірний елемент *b* – одним випадково обраним твірним елементом дочірньої множини Y_2.

Можливими результатами генерації є

$$W_z = \{\underbrace{(cdefg),(dcefg),(ecdfg)}_{result\ of\ replacing\ (ab)},$$
$$\underbrace{(gfced),(gfdec),(gfedc)}_{result\ of\ replacing\ (ba)}\}.$$

2.3.2 Оцінка складності алгоритму випадкової генерації *k*-множин

Оскільки алгоритм випадкової генерації *k*-множин *Gen_Random_k-set* є модифікацією «повного» алгоритму *Gen_k-set*, для оцінки його складності може використовуватися формула (2.13). Для опису складності алгоритму *Gen_Random_k-set* за допомогою формули (2.13) достатньо замінити в ній *Card(Y)* на *V(Y)*. Таким чином, складність алгоритму випадкової генерації *k*-множин *Gen_Random_k-set* можна оцінити як

$$\sum_{i=0}^{k}\sum_{j=1}^{\eta_i} O(Y_{ij}) + \sum_{i=0}^{k-1}(Card(P^i)\prod_{j=1}^{\eta_{i+1}} V(Y_{(i+1)\,j})),$$

де $O(Y_{ij})$ – складність алгоритму генерації кожної базової комбінаторної множини Y_{ij} (i – рівень ієрархії базової комбінаторної множини в *k*-множині, j – порядковий номер базової комбінаторної множини на *i*-му рівні *k*-множини);

$P^i = \Gamma_i \circ \Gamma_{i-1} \circ ... \circ \Gamma_0(z)$ – проміжкова *k*-множина, що виступає батьківською множиною при виконанні операції *n*-заміщення на *i*-му рівні *k*-множини.

2.4 Перестановки з частково заданою сигнатурою

У даному пункті введено нову комбінаторну множину – перестановки з частково заданою сигнатурою, що являють собою узагальнений випадок перестановок з заданими підйомами та спусками, описаних в роботах F. Ruskey та N.G. De Brujin. Наведено математичний опис запропонованого класу перестановок, отримано формулу для обчислення їх кількості, побудовано

метод та алгоритм їх генерації, що базуються на використанні узагальнених методу та алгоритму комбінаторної генерації, розглянутих в пункті 2.1. Наведено приклад роботи запропонованого алгоритму та оцінено його обчислювальну складність.

2.4.1 Математичний опис перестановок з частково заданою сигнатурою

Нехай задана множина твірних елементів $A=\{a_1,a_2,...,a_n\}$, $a_1<a_2<...<a_n$, $a_i\in R^1$, $i\in J_n$, $J_n=\{1,2,...,n\}$. Множину всіх можливих перестановок з елементів $a_1,a_2,...,a_n$ позначимо P_n. Для будь-якої перестановки $\pi=(\pi_1,\pi_2,...,\pi_n)\in P_n, \pi_j\in A, j\in J_n$ справедливі відношення:

$$\pi_1\rho_1\pi_2\rho_2...\rho_{n-1}\pi_n, \rho_i\in\{<,>\}, i\in J_{n-1}. \quad (2.17)$$

Інакше кажучи,

$$\pi_i<\pi_{i+1} \quad (2.18)$$

або

$$\pi_i>\pi_{i+1}. \quad (2.19)$$

Послідовність $\rho(\pi)=(\rho_1,\rho_2,...,\rho_{n-1})$ відповідає сигнатурі перестановки, описаній в роботі F. Ruskey [137]. Далі під сигнатурою перестановки будемо розуміти $\rho(\pi)$.

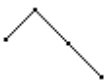
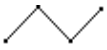
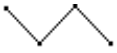
Лема 2.2. У множині P_n усіх можливих перестановок з n елементів існує 2^{n-1} різних сигнатур $\rho(\pi)$.

Доведення леми випливає з існування взаємно однозначної відповідності між множиною всіх послідовностей $\rho_1, \rho_2, ..., \rho_{n-1}$ і множиною всіх двійкових послідовностей з $n-1$ елементів.

Сигнатура $\rho(\pi)$ перестановки $\pi \in P_n$ можна зобразити за допомогою діаграми, що складається з $n-1$ сегменту. Виконанню нерівності (2.18) відповідає ділянка підйому, а виконанню нерівності (2.19) – ділянка падіння.

Приклад 2.5. Нехай множина перестановок P_4 утворена елементами $\{1,2,3,4\}$. Тоді, згідно лемі 2.2, існує всього вісім різних сигнатур в перестановках множини P_4. Діаграми, що представляють собою сигнатури, і відповідні їм перестановки наведено в таблиці 2.1.

Діаграми і перестановки множини P_4 *Таблиця 2.1*

Діаграма	Перестановки	Діаграма	Перестановки
	(1,2,3,4)		(4,3,2,1)
	(1,2,4,3) (1,3,4,2) (2,3,4,1)		(3,4,2,1) (2,4,3,1) (1,4,3,2)
	(3,2,1,4) (4,3,1,2) (4,2,1,3)		(4,1,2,3) (2,1,3,4) (3,1,2,4)
	(1,3,2,4) (1,4,2,3) (2,4,1,3) (2,3,1,4) (3,4,1,2)		(4,2,3,1) (3,2,4,1) (3,1,4,2) (4,1,3,2) (2,1,4,3)

Довільну підпослідовність $\rho(\pi)$, тобто послідовність $r(\pi) = (\rho_{i_1}, \rho_{i_2}, ..., \rho_{i_k})$, $\{i_1, i_2, ..., i_k\} \subset J_{n-1}$, назвемо частково заданою сигнатурою перестановки $\pi \in P_n$. «Повна» сигнатура $\rho(\pi)$ визначає відношення елементів π_i і π_{i+1} для всіх позицій $i \in J_{n-1}$, частково задана сигнатура $r(\pi)$ – лише для деяких $i \in \{i_1, i_2, ..., i_k\} \subseteq J_{n-1}$.

Зафіксуємо обрану частково задану сигнатуру $r(\pi)$ перестановки $\pi \in P_n$. Їй можна поставити у відповідність частину множини спуску перестановки [65]. Множина $D(\pi)$ спуску перестановки π визначається в [65] як

$$D(\pi) = \{i \,|\, \pi_i > \pi_{i+1}\}, i \in J_{n-1}.$$

Визначимо множину $\bar{D}(\pi) = J_{n-1} \setminus D(\pi)$ як

$$\bar{D}(\pi) = \{i \,|\, \pi_i < \pi_{i+1}\},\ i \in J_{n-1}.$$

Нас цікавить частина $D(\pi)$, яка задає сигнатуру для позицій $i \in \{i_1, i_2, ..., i_k\}$. Позначимо її через $D'(\pi)$:

$$D'(\pi) = \{i \,|\, \pi_i > \pi_{i+1}\} \subseteq D(\pi), i \in \{i_1, i_2, ..., i_k\}.$$

Тоді множину $\bar{D}'(\pi) = \{i_1, i_2, ..., i_k\} \setminus D'(\pi)$ можна визначити у вигляді

$$\bar{D}'(\pi) = \{i \,|\, \pi_i < \pi_{i+1}\} \subseteq \bar{D}(\pi), i \in \{i_1, i_2, ..., i_k\}.$$

Далі множину $D'(\pi)$ будемо називати множиною спуску, а $\bar{D}'(\pi)$ – множиною підйому.

Частково задана сигнатура $r(\pi)$ та множини $D'(\pi)$ і $\bar{D}'(\pi)$ однозначно визначають один одного. Наприклад, якщо $r(\pi) = (\rho_2, \rho_4, \rho_5) = (>, <, >)$, то $D'(\pi) = \{2,5\}$ і $\bar{D}'(\pi) = \{4\}$. Це твердження справедливе також для «повної» сигнатури $\rho(\pi)$ як окремого випадку $r(\pi)$. Наприклад, в останньому рядку таблиці *2.1* множина спуску перестановок, що відповідає діаграмі зліва, містить один елемент: $D(\pi) = \{2\}$, а $\bar{D}(\pi) = \{1,3\}$. Для перестановок, відповідних діаграмі праворуч, $D(\pi) = \{1,3\}$, $\bar{D}(\pi) = \{2\}$.

Перестановка задовольняє частково заданій сигнатурі $r(\pi)$ тоді, коли для всіх $i \in \{i_1, i_2, ..., i_k\}$ відношення π_i і π_{i+1} відповідає заданому в $r(\pi)$. Відношення π_i і π_{i+1} для $i \notin \{i_1, i_2, ..., i_k\}$ при цьому може бути будь-яким

$$\rho_i \in \{<,>\},\ i \notin \{i_1, i_2, ..., i_k\}.$$

Лема 2.3. Щоб перестановка $\pi \in P_n$ задовольняла частково заданій сигнатурі $r(\pi)$, заданій для k позицій $i \in \{i_1, i_2, ..., i_k\}$, необхідно і достатньо, щоб вона задовольняла одній з 2^{n-k-1} «повних» сигнатур $\rho(\pi)$, що збігаються з $r(\pi)$ для $i \in \{i_1, i_2, ..., i_k\}$.

Доведення. Через те, що відношення елементів π_i і π_{i+1} при $i \notin \{i_1, i_2, ..., i_k\}$ може бути будь-яким, то в перестановках, що задовольняють частково заданій сигнатурі $r(\pi)$, на позиціях $i \notin \{i_1, i_2, ..., i_k\}$ може виконуватися як нерівність $\pi_i < \pi_{i+1}$, так і $\pi_i > \pi_{i+1}$. Це означає, що, якщо частково задана сигнатура $r(\pi)$ визначена для k позицій з $n-1$ можливих, то для інших $n-1-k$ позицій відношення сусідніх елементів π_i, π_{i+1} може бути будь-яким. Отже, для кожної з решти $n-1-k$ позицій $i \notin \{i_1, i_2, ..., i_k\}$ можна зафіксувати будь-яке відношення $\rho_i \in \{<,>\}$, отримавши таким чином 2^{n-k-1} «повних» сигнатур $\rho(\pi)$. У перестановках, що відповідають кожній такій сигнатурі $\rho(\pi)$, відношення π_i і π_{i+1} при $i \in \{i_1, i_2, ..., i_k\}$ буде відповідати заданому в $r(\pi)$.

Якщо в частково заданій сигнатурі $r(\pi)$ задано всі позиції $i \in \{1, 2, ..., n-1\}$, то перестановки з частково заданою сигнатурою перетворюються в перестановки з повністю заданою сигнатурою $\rho(\pi)$, або в перестановки з заданими підйомами та спусками [132]. Окремими випадками таких перестановок є унімодальні перестановки [140] при $\rho(\pi) = (>,>,...,>,<,...,<)$ або $\rho(\pi) = (<,<,...,<,>,...,>)$ (тобто коли спочатку йдуть тільки спуски, а потім – тільки підйоми, або навпаки), а також альтернативні перестановки [138] при $\rho(\pi) = (>,<,>,<,...)$ або $\rho(\pi) = (<,<,...,<,>,...,>)$ (тобто коли спуски і підйоми чергуються).

2.4.2 Перечислення перестановок з частково заданою сигнатурою

Розглянемо задачу перечислення перестановок з частково заданою сигнатурою $r(\pi)$.

З леми 2.3 випливає, що кількість перестановок, що відповідають частково заданій сигнатурі $r(\pi)$, дорівнює сумі кількості перестановок, що відповідають кожній з $M = 2^{n-k-1}$ «повних» сигнатур $\rho^j(\pi),\ j \in J_M$, що відповідають $r(\pi)$. Тоді для частково заданої сигнатурі $r(\pi)$ необхідно знайти всі можливі відповідні їй «повні» сигнатури $\rho^1(\pi), \rho^2(\pi), ..., \rho^M(\pi)$, потім визначити кількість перестановок, що відповідають кожній з них, і скласти.

Розглянемо задачу перечислення перестановок $\pi \in P_n$ з заданою сигнатурою $\rho^j(\pi),\ j \in J_M$. Задання сигнатури $\rho^j(\pi)$ еквівалентне заданню множини спуску $S^j \subseteq J_{n-1}$. Іншими словами, для перечислення перестановок $\pi \in P_n$ з заданою сигнатурою $\rho^j(\pi)$ необхідно підрахувати число $N(\rho^j(\pi))$ всіх перестановок $\pi \in P_n$, таких, що $D(\pi) = S^j$.

Результати, отримані в [65], дозволяють за допомогою принципу включень-виключень визначити кількість перестановок, що відповідають заданій множині спуску, тобто перестановок із заданою «повною» сигнатурою (що еквівалентні перестановкам з заданими підйомами та спусками). Дотримуючись [65], для $S^j = \{s_1^j, s_2^j, ..., s_k^j\} \subseteq J_{n-1}$ позначимо через $\beta_n(S) = N(\rho^j(\pi))$ кількість перестановок $\pi \in P_n$, для яких S є множиною спуску.

В [65] отримана наступна формула для обчислення $\beta_n(S)$:

$$N(\rho^j(\pi)) = \beta_n(S) = \sum_{1 \le i_1 < i_2 < ... < i_z \le k} (-1)^{k-z} \binom{n}{s_{i_1}^j, s_{i_1}^j - s_{i_1}^j, ..., n - s_{i_z}^j}, \qquad (2.20)$$

де

$$\binom{n}{n_1, n_2, ..., n_k} = \frac{n!}{n_1! \cdot n_2! \cdot ... \cdot n_k!}.$$

Наведені міркування роблять справедливою наступну лему.

Лема 2.4. Нехай $\rho^j(\pi^0)$ – сигнатура перестановки $\pi^0 \in P_n$, причому $D(\pi^0) = S^j = \{s_1^j, s_2^j, ..., s_k^j\} \subseteq J_{n-1}, j \in J_{n-k-1}$. Кількість $N(\rho^j(\pi^0))$ всіх перестановок $\pi \in P_n$, що мають сигнатуру $\rho^j(\pi) = \rho^j(\pi^0)$, дорівнює $\beta_n(S)$, тобто $N(\rho^j(\pi^0)) = \beta_n(S^j)$, і визначається за формулою **(**2.20**)**.

Доказ. Визначимо бінарне відношення R в множині P_n таким чином. Два елементи $\pi^1, \pi^2 \in P_n$ перебувають у відношенні R, якщо вони мають одну і ту саму сигнатуру $\rho^j(\pi^1) = \rho^j(\pi^2)$. Для відношення R неважко перевірити справедливість властивостей рефлексивності, симетричності і транзитивності, що дозволяє зробити висновок про те, що R є відношенням еквівалентності в множині P_n.

Тоді всі елементи $\pi \in P_n$, що мають одну і ту саму сигнатуру, утворюють клас еквівалентності $[\pi]$ по відношенню R. Множина всіх класів еквівалентності становить фактор-множину $\Pi = P_n / R$ множини P_n по відношенню R.

Нехай $\Lambda : P_n \to \Pi$ – фактор-відображення на множині P_n. Розглянемо задачу реалізації фактор-відображення Λ в явному вигляді. Розв'язання цієї задачі зводиться до генерації всіх перестановок $\pi \in P_n$ з сигнатурою $\rho^j(\pi) = \rho^j(\pi^0)$ для будь-якої перестановки $\pi^0 \in P_n$.

Склавши кількість перестановок, що мають сигнатуру $\rho^1(\pi), \rho^2(\pi), ..., \rho^M(\pi)$, отримаємо кількість перестановок, що відповідають частково заданій сигнатурі $r(\pi)$

$$N(r(\pi)) = \sum_{j=1}^{M} N(\rho^j(\pi)),\ M = 2^{n-k-1}. \qquad \textbf{(}2.21\textbf{)}$$

Побудуємо метод та алгоритм генерації перестановок $\pi \in P_n$ з частково заданою сигнатурою $r(\pi)$.

2.4.3 Генерація перестановок з частково заданою сигнатурою

Необхідно згенерувати всі перестановки $\pi \in P_n$, що відповідають частково заданій сигнатурі $r(\pi)$, в кількості $N(r(\pi))$, що визначається за формулою **(**2.21**)**. Для генерації всіх перестановок з частково заданою сигнатурою $r(\pi)$ використаємо описаний в пункті 2.1 алгоритм *GenBase*.

Використання алгоритму *GenBase* потребує задання множини параметрів *p*, до якої входить процедура *GetF* та специфічні для конкретної комбінаторної множини параметри. Для перестановок з частково заданою сигнатурою цими параметрами є:

- множина твірних елементів $A = \{a_1, a_2, ..., a_n\},\ a_1 < a_2 < ... < a_n$;
- множини $D'(\pi)$ і $\overline{D}'(\pi)$, що визначають частково задану сигнатуру $r(\pi)$.

Для зручності, частковий кортеж t^i з пункту 0 будемо називати частковою перестановкою і позначати як π^i, $\pi^i = (\pi_1, \pi_2, ..., \pi_i)$, $\pi_i \in A$, $i \in J_n^0$, $J_n^0 = \{0,1,2,...,n\}$. При цьому π^0 – порожня часткова перестановка (відповідає порожньому кортежу $t^0 = ()$ з пункту 0), $\pi^n = \pi \in P_n$ – результат генерації.

Розглянемо принцип побудови процедури *GetF*. Для перестановок з частково заданою сигнатурою, кортеж F^i формується з твірних елементів, що задовольняють наступним умовам:

1) кожен елемент-кортежу F^i, що стане наступним елементом π_{i+1} поточної часткової перестановки π^i, повинен відрізнятися від всіх попередніх елементів часткової перестановки

$$\pi_{i+1} \neq \pi_j, \ \forall j \in J_i; \tag{2.22}$$

2) якщо частково задана сигнатура $r(\pi)$ передбачає спуск в позиції i (позиція i належить множині $D'(\pi)$), то π_{i+1} повинен бути менше π_i

$$i \in D'(\pi) \Rightarrow \pi_{i+1} < \pi_i; \tag{2.23}$$

якщо частково задана сигнатура $r(\pi)$ передбачає підйом в позиції i (позиція i належить множині $\overline{D}'(\pi)$), то π_{i+1} повинен бути більше π_i

$$i \in \overline{D}'(\pi) \Rightarrow \pi_{i+1} > \pi_i; \tag{2.24}$$

якщо частково задана сигнатура $r(\pi)$ не задана для i-ої позиції, то елемент π_{i+1} може бути як більше, так і менше π_i, і на π_{i+1} не накладаються ніякі додаткові обмеження

$$i \notin \overline{D}(\pi), i \notin \overline{D}'(\pi) \Rightarrow \pi_{i+1} \rho_i \pi_i, \rho_i \in \{<, >\}; \tag{2.25}$$

3) якщо, починаючи з позиції $i+1$, частково задана сигнатура $r(\pi)$ передбачає m спусків підряд, то елемент π_{i+1} повинен обиратися таким чином, щоб існувало щонайменше m «вільних» елементів множини A, тобто елементів множини A, що не містяться в поточній частковій перестановці π^i, і є меншими за π_{i+1}

$$\{i+1,i+2,...,i+m\} \subseteq D'(\pi) \Rightarrow \exists L = \{a_{l_1}, a_{l_2}, ..., a_{l_m}\} \subseteq A: \forall a_{l_j} < \pi_{i+1}, \quad a_{l_j} \neq \pi_k, \tag{2.26}$$

де $j \in J_m$, $k \in J_i$, $m \in J_{n-1}$;

і навпаки: якщо, починаючи з позиції $i+1$, частково задана сигнатура $r(\pi)$ передбачає m підйомів підряд, то елемент π_{i+1} повинен обиратися таким чином, щоб існувало щонайменше m елементів множини A, що не містяться в поточній частковій перестановці π^i, і є більшими за π_{i+1}

$$\{i+1,i+2,...,i+m\} \subseteq \overline{D}'(\pi) \Rightarrow \exists L = \{a_{l_1}, a_{l_2}, ..., a_{l_m}\} \subseteq A: \forall a_{l_j} > \pi_{i+1}, \; a_{l_j} \neq \pi_k, \tag{2.27}$$

де $\forall j \in J_m$, $k \in J_i$, $m \in J_{n-1}$;

якщо частково задана сигнатура $r(\pi)$ не визначена для позиції $i+1$, то елемени, що слідують за π_{i+1}, можуть бути як більше, так і менше π_{i+1}, і ніякі додаткові обмеження (за винятком обмеження унікальності елементу) не накладаються.

На рівнях $i \in J_{n-2}$ у кортеж F^i входять усі твірні елементи, що задовольняють умовам (2.22)-(2.27). На рівнях $i = 0$ и $i = n-1$ умови формування кортежу F^i дещо змінюються.

На нульовому рівні умови (2.22)-(2.25) не перевіряються, тому що перестановка π^0 не містить елементів. У цьому випадку кортеж F^0 формується таким чином:

– якщо частково задана сигнатура $r(\pi)$ на початку (тобто починаючи с позиції $i=1$) передбачає m спусків, то кожен елемент-кортеж F^0 повинен бути таким, щоб існувало m твірних елементів , менших за нього. Таким чином, кортеж F^0 містить останні $n-m$ твірні елементи

$$\{1,2,...,m\} \subseteq D'(\pi) \Rightarrow F^0 = \{a_{m+1}, a_{m+2}, ..., a_n\}; \tag{2.28}$$

– якщо частково задана сигнатура $r(\pi)$ передбачає на початку m підйомів, то кожен елемент-кортеж F^0 повинен бути таким, щоб існувало щонайменше m твірних елементів, більших за нього. Таким чином, кортеж F^0 містить перші $n-m$ елементів твірної множини:

$$\{1,2,...,m\} \subseteq \overline{D}'(\pi) \Rightarrow F^0 = \{a_1, a_2, ..., a_{n-m}\}; \quad \textbf{(2.29)}$$

– якщо частково задана сигнатура $r(\pi)$ не встановлює відношення сусідніх елементів для позиції $i=1$, то кортеж F^0 містить усі твірні елементи:

$$1 \notin D'(\pi), 1 \notin \overline{D}'(\pi) \Rightarrow F^0 = \{a_1, a_2, ..., a_n\}. \quad \textbf{(2.30)}$$

На рівні $n-1$ кортеж F^{n-1} містить єдиний твірний елемент, що не входить до π^{n-1}. Такий елемент «автоматично» задовольняє частково заданій сигнатурі $r(\pi)$, тому що його існування перевіряється на рівні $n-2$ в умовах (2.26), (2.27). Це значить, що умови (2.22)-(2.27) не перевіряються.

Величина m на кожному рівні $i \in J_{n-2}^0$ приймає значення, що залежить від $r(\pi)$. Для зручності на рівнях $i \in J_{n-2}^0$ через m^i позначимо *кількість слідуючих один за одним підйомів або спусків в частково заданій сигнатурі, починаючи з наступної позиції* $i+1$. Значення m^i обчислюються лиш тоді, коли позиція $i+1$ належить або до $D'(\pi)$, або до $\overline{D}'(\pi)$, тобто m^i не обчислюється, коли позиція $i+1$ не задана.

Значення m^i обчислюються таким чином:

– якщо частково задана сигнатура $r(\pi)$ с позиції $i+1$ передбачає m^i спусків, то m^i – кількість таких спусків, що слідують одне за одним, починаючи з позиції $i+1$

$$i+1 \in D(\pi) \Rightarrow m^i = \max(t : i+t \in D(\pi)), t \in J_{n-i-2}; \quad \textbf{(}2.31\textbf{)}$$

– якщо частково задана сигнатура $r(\pi)$, починаючи з позиції $i+1$, передбачає m^i підйомів, то m^i – кількість таких підйомів, що слідують одне за одним, починаючи з позиції $i+1$

$$i+1 \in \overline{D}(\pi) \Rightarrow m^i = \max(t : i+t \in \overline{D}(\pi)), t \in J_{n-i-2}. \quad \textbf{(}2.32\textbf{)}$$

Для частково заданої сигнатури $r(\pi)$ обчислимо всі m^i за допомогою функції *GetM*. Вихідні дані *GetM* – потужність множини твірних елементів n, частково задана сигнатура $r(\pi)$, якій відповідають множини $D'(\pi)$ и $\overline{D}'(\pi)$. Функція *GetM* наведена нижче.

def *GetM*(i, n, $D'(\pi)$, $\overline{D}'(\pi)$)
for i in {0, 1, ..., n - 2}
    if $\{i+1\} \subseteq D'(\pi)$
        $m^i := \max(t : i+t \in D'(\pi))$
    else
        $m^i := \max(t : i+t \in \overline{D}'(\pi))$

Обчисливши за допомогою наведеної вище функції *GetM* всі значення m^i на кожному рівні $i \in J_{n-2}^0$, в умовах (2.26)-(2.29) під значенням m будемо мати на увазі значення m^i для кожного рівня $i \in J_{n-2}^0$.

Опишемо тепер роботу процедури *GetF_updown*, що передається в алгоритм *GenBase* у множині параметрів p та використовується для формування кортежу F^i. Функція приймає на вхід часткову перестановку π^i

(що відповідає частковому кортежу t^i), а також множину параметрів *p*, що містить:

- множину твірних елементів $A=\{a_1,a_2,...,a_n\}$, $a_1<a_2<...<a_n$;
- множини $D'(\pi)$ і $\overline{D}'(\pi)$, що визначають частково задану сигнатуру $r(\pi)$;
- значення $m^i, i\in J_{n-2}^0$, тобто множину $\{m_0,m_1,...,m_{n-2}\}$.

На початку, коли $i=0$, способом (2.28)-(2.30) (в залежності від $r(\pi)$) формується і повертається кортеж F^0.

Далі, коли $i\neq 0$, кортеж F^i формується в циклі по всім твірним елементам $a_j\in A$, що не входять до поточної часткової перестановки π^i. Булева змінна *condition2*, що відображає виконання умов (2.23)-(2.25), приймає значення *True*, якщо умови (2.23)-(2.25) виконані, інакше *False*. Подібним чином, змінна *condition3* відповідає за виконання умов (2.26)-(2.27). Як було відмічено раніше, для рівня $n-1$ умови (2.22)-(2.27) завжди будуть задовольнятися, тому вони не перевіряються. Елементи $a_j\in A$, що задовольняють описаним вище умовам, формують кортеж F^i. Через те, що елементи $a_j\in A$ перебираються послідовно, а множина A строго впорядкована по зростанню, кортеж F^i також є строго впорядкованим по зростанню елементів.

def *GetF_updown* (π^i, { n, $D'(\pi)$, $\overline{D}'(\pi)$, $A=\{a_1,a_2,...,a_n\}$, $\{m_0,m_1,...,m_{n-2}\}$ })

if $i==0$

if $1\in D'(\pi)$

return $\{a_{m^0+1},a_{m^0+2},...,a_n\}$

if $1\in\overline{D}'(\pi)$

return $\{a_1,a_2,...,a_{n-m^0}\}$

if $1\notin D'(\pi),1\notin\overline{D}'(\pi)$

return $\{a_1,a_2,...,a_n\}$

$F^i=\varnothing$

for ($a\in A, a\neq\pi^i_j$, $j=1,2,...,i$)

if ($i\in D(\pi)$ **and** $a<\pi^i_i$**) or (**$i\in\overline{D}'(\pi)$ **and** $a>\pi^i_i$**) or** $i\notin D'(\pi), i\notin\overline{D}'(\pi)$

condition2 = True

else

condition2 = False

if $i+1\in D'(\pi)$

$condition3 = \exists L=\{a_{l_1},a_{l_2},...,a_{l_{m^i}}\}\subseteq A: \forall a_{l_j}<\pi_{i+1},\ a_{l_j}\neq\pi_k,$

$\forall j\in J_{m^i},\ \forall k\in J_i$

if $i+1\in\overline{D}'(\pi)$

$condition3 = \exists L=\{a_{l_1},a_{l_2},...,a_{l_{m^i}}\}\subseteq A: \forall a_{l_j}>\pi_{i+1},$

$a_{l_j}\neq\pi_k,\ \forall j\in J_{m^i}, \forall k\in J_i$

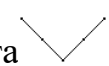
if $i+1\in D'(\pi), i+1\in\overline{D}'(\pi)$

condition3 = True

if ($i==n-1$) **or** (*condition2* **and** *condition3*)

$F^i = F^i\cup a$

return F^i

Приклад 2.6. На рисунку 2.6 представлено рекурсивне дерево роботи алгоритму *GenBase*, збудоване в процесі генерації перестановок з елементів $A=\{1,2,3,4,5\}$ при частково заданій сигнатурі, що визначається множинами $D'(\pi)=\{1\}$ та $\overline{D}'(\pi)=\{3,4\}$ (позиція $i=2$ не визначена). Лема 2.3 в даному випадку звучить так: для того, щоб перестановка $\pi\in P_n$ задовольняла частково заданій сигнатурі $r(\pi)$, визначеній для $k=3$ позицій $i\in\{1,3,4\}$, необхідно і достатньо, щоб вона задовольняла одному з 2 «повних» сигнатур $\rho(\pi)$, що співпадають з $r(\pi)$ для $i\in\{1,3,4\}$. Такі «повні» сигнатури повинні мати спуск на позиції 1 та підйом на позиціях 3,4 і відповідають діаграмам  та  .

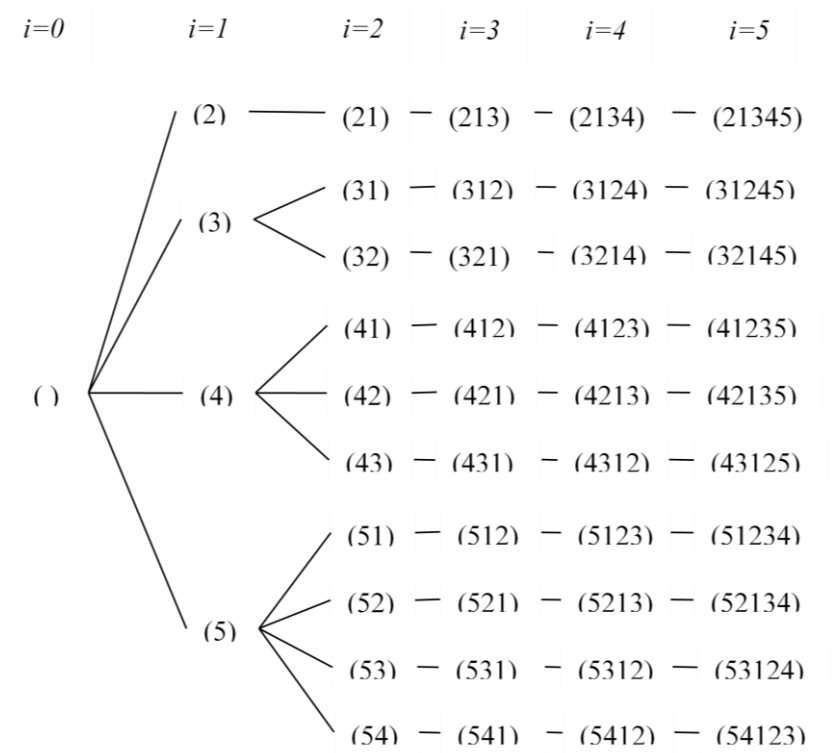


Рисунок 2.6. Рекурсивне дерево роботи алгоритму *GenBase* при генерації перестановок з елементів $\{1,2,3,4,5\}$ та заданих множинах спуску $D'(\pi)=\{1\}$ та підйому $\overline{D}'(\pi)=\{3,4\}$.

Застосуємо алгоритм *GetM* для підрахунку значень m^i. У результаті отримаємо

- $m^0=1$
- m^1 – не обчислюється (тому, що позиція 2 не визначена в частково заданій сигнатурі),
- $m^2=2$
- $m^3=1$

Наведемо кортежі F^i на рівнях $i=0,1$.

На рівні $i=0$ $\pi^0=(\ )$, $1\in D'(\pi)$, кортеж $F^0=(a_{1+1},...,a_5)=(2,3,4,5)$ у відповідності з **(**2.29**)**.

Через те, що множина F^0 містить 4 елементи, алгоритм «розгалужується», отримуючи на рівні $i=1$ часткові перестановки $\pi^1=(2), \pi^1=(3), \pi^1=(4), \pi^1=(5)$.

Розглянемо приклад формування кортежу F^1 для часткової перестановки $\pi^1=(3)$. Умова (2.22) приймає вигляд $\pi_2 \neq 3$. Так як $1 \in D'(\pi)$, перевіряється умова (2.23), шо приймає вигляд $\pi_2 < \pi_1$. Оскільки наступна позиція $i=2$ не визначена, ніякі додаткові умови не перевіряються. Утворюючими елементами, що задовольняють описаним вище умовам і можуть стати на місце наступного елемента π_2 поточної часткової перестановки, є елементи $\{1,2\}$.

На цьому ж рівні для перестановки $\pi^1=(4)$ множина $F^1=\{1,2,3\}$.

Так як для $\pi^1=(1)$ множина F^1 містить 2 елементи, то знову відбувається розгалуження рекурсивного дерева (backtracking tree). На рівнях $i=2,3,4$ множини F^2, F^3 та F^4 містять по одному елементу, тому алгоритм *GenBase* не здійснює розгалужень. На рівні $i=5$ довжина поточної перестановки дорівнює 5, що означає, що поточна перестановка є повною, і алгоритм додає $\pi=\pi^5$ у множину всіх згенерованих перестановок *T*.

Підрахувавши кількість всіх перестановок с частково заданою сигнатурою за формулою (2.21), отримаємо в результаті 10, що співпадає з кількістю перестановок, отриманих в результаті роботи алгоритму.

2.4.4 Оцінка складності алгоритму генерації перестановок з частково заданою сигнатурою

В підпункті 2.2.4 складність алгоритму *GenBase* була оцінена як кількість рекурсивних викликів *GenBase*, що відбувалися з моменту виклику алгоритма до закінчення його роботи. На кожному рівні рекурсії в поточну часткову перестановку додається один елемент, тому для генерації кожної з N

перестановок $\pi^n = \pi \in P_n$ *GenBase* викликається не більше ніж *n* раз. Тому, як зазначено в підпункті 2.2.4, складність алгоритму *GenBase* не перевищує nN.

Однак алгоритм *GetF_updown* формування F^i є досить складним, і тому важливо показати, що не виникає ситуацій, коли алгоритм повертає порожній кортеж F^i. Виникнення таких ситуацій значило б появу тупиків (тобто недобудованих вершин довжини, меншої *n*) у рекурсивному дереві, що будується у процесі роботи алгоритму *GenBase*. Це могло б значно підвищити обчислювальну складність алгоритму. Тому важливо показати, що таких ситуацій не виникає.

Теорема 2.1. Алгоритм *GenBase*, використаний для генерації перестановок з частково заданою сигнатурою, будує рекурсивне дерево (backtracking tree [27]), кожним листом (leaf) якого є перестановка π, що задовольняє частково заданій сигнатурі $r(\pi)$.

Доведення. Якщо лист дерева є перестановкою π довжини n, то це значить, що всі елементи перестановки π^n задовольняють умовам (2.22)-(2.27) і тому π^n є перестановкою з частково заданою сигнатурою $r(\pi)$. Покажемо, що рекурсивне дерево не містить тупиків, тобто листів, де довжина π^i менше n. Тупик на рівні $i \in J_{n-1}^0$ утворюється, коли кортеж F^i є порожнім, тобто коли жоден з твірних елементів не задовольняє водночас всім умовам (2.22)-(2.27). Показавши, що кортеж F^i на кожному рівні $i \in J_{n-1}^0$ містить щонайменше 1 елемент, доведемо, що рекурсивне дерево, що будується під час роботи алгоритму *GenBase*, не має тупиків.

На рівні i завжди існує $n-i$ твірних елементів , що не входять до π^i. Таким чином, завжди можна знайти $n-i$ твірних елементів, що задовольняють умові (2.22).

Через те, що твірні елементи не дорівнюють один одному, кожен з $n-i$ твірних елементів , що не входять в π^i, або більше π_i, або менше за нього. Для

того, щоб твірний елемент задовольняв умовам (2.23) та (2.24), необхідно, щоб він був відповідно менше або більше π_i у залежності від заданої сигнатури $r(\pi)$.

З формул (2.26) та (2.27) випливає, що на рівні $i-1$ елемент π_i обирається таким чином, щоб існувало щонайменше $m^{i-1} \geq 1$ твірних елементів, що не містяться в π^{i-1} та є відповідно меншими або більшими π_i. Звідси випливає, що на рівні i завжди існує як мінімум $m^{i-1} \geq 1$ твірних елементів, що задовольняють умовам, описаним в (2.26) та (2.27).

Позначимо через $C^i = \{a_{c_1}, a_{c_2}, ..., a_{c_p}\}$, $c_1 < c_2 < ... < c_p$, $\{c_1, c_2, ..., c_p\} \subseteq J_n$ строго впорядковану за зростанням множину твірних елементів, що задовольняють умовам **(**2.22**)-(**2.24**)** на рівні i. Для того, щоб елементи множини C^i мали можливість бути включеними до кортежу F^i, вони повинні задовольняти умовам **(**2.26**)** та **(**2.27**)**.

Для виконання умов **(**2.26**)** та **(**2.27**)** для елементу $a_{c_k}, k \in J_p$, необхідно, щоб існувало щонайменше m^i елементів множини C^i, відповідно менших або більших a_{c_k} (у залежності від $r(\pi)$). Використовуючи термінологію [65] і дещо модифікувавши її, відмітимо, що в залежності від частково заданої сигнатури $r(\pi)$ елемент π_{i+1}, кандидатом на який є твірний елемент a_{c_k}, може бути:

- спуском, якщо $\pi_i > \pi_{i+1} > \pi_{i+2}$, тобто $\{i, i+1\} \subseteq D(\pi)$;
- піком, якщо $\pi_i < \pi_{i+1} > \pi_{i+2}$, тобто $i \in \overline{D}(\pi), i+1 \in D(\pi)$;
- підйомом, якщо $\pi_i < \pi_{i+1} < \pi_{i+2}$, тобто $\{i, i+1\} \subseteq \overline{D}(\pi)$;
- долиною, якщо $\pi_i > \pi_{i+1} < \pi_{i+2}$, тобто $i \in D(\pi), i+1 \in \overline{D}(\pi)$.

У даному випадку для спуску та піку, починаючи з позиції $i+1$, може слідувати m^i позицій, що належать до множини спуску, тобто

$\{i, i+1, ..., i+m^i\} \subseteq D(\pi)$ для спуску, $i \in \overline{D}(\pi), \{i+1, ..., i+m^i\} \subseteq D(\pi)$ для піку. Для підйому та долини, починаючи з позиції $i+1$, може слідувати m^i позицій, що належать до множини підйому, тобто $\{i, i+1, ..., i+m^i\} \subseteq \overline{D}(\pi)$ для підйому, $i \in D(\pi), \{i+1, ..., i+m^i\} \subseteq \overline{D}(\pi)$ для долини.

Таким чином, крім умови (2.22), для спуску повинні перевірятися умови (2.23) та (2.26), для піку – (2.24) та (2.26), для підйому – (2.24) та (2.27), для долини – (2.23) та (2.27).

Якщо π_{i+1} – підйом або спуск, то існування $m^i = m^{i-1} - 1$ елементів множини C^i, відповідно більших або менших π_{i+1}, перевіряється ще на попередньому рівні $i-1$ при перевірці умов (2.27) та (2.26) відповідно.

Якщо π_{i+1} – пік, то до множини C^i потрапляють твірні елементи, що задовольняють умові (2.24), тобто є більшими за π_i. Для виконання умови (2.26) для елементу a_{c_k} потребується, щоб існувало як мінімум m^i елементів множини C^i, що є меншими за a_{c_k}. Якщо множина C^i містить тільки 1 елемент a_{c_1}, то для інших $n-i-1$ елементів, що не входять до π^i, умова (2.24) не виконується, тобто вони є меншими за π_i, і тому також меншими за a_{c_1}. Якщо у множині C^i міститься $p>1$ елементів, то існує $n-i-p$ твірних елементів, що не входять до π^i і не належать до множини C^i (тобто не задовольняють умові (2.24)).

Якщо $n-i-p \geq m^i$, то умові (2.26) задовольняють усі елементи множини C^i, тому що для кожного a_{c_k} завжди можна знайти $n-i-p$ твірних елементів, що є меншими за a_{c_k} і не входять до поточної часткової перестановки π^i.

Якщо $n-i-p<m^i$, то умові (2.26) задовольняють не всі елементи C^i, а тільки ті, для яких кількість твірних елементів, що не входять до π^i та є меншими за a_{c_k}, дорівнює m^i. Таким чином, необхідно мати m^i елементів, менших a_{c_k}, а поза множиною C^i їх тільки $n-i-p$, що значить, що необхідно мати що $m^i-(n-i-p)$ елементів множини C^i, що є меншими за a_{c_k}. Звідси умова (2.26) виконується для останніх $p-(m^i-(n-i-p))=n-i-m^i$ елементів C^i. При цьому m^i завжди менше $n-i$ через те, що кількість відношень «більше» або «менше» між $n-i$ елементами $n-i-1\geq m^i$.

Якщо π_{i+1} – долина, то до множини C^i потрапляють твірні елементи, що задовольняють (2.23), тобто є меншими за π_i. Для виконання умови (2.27) для елементу a_{c_k} потрібно, щоб існувало як мінімум m^i елементів множини C^i, що є більшими за a_{c_k}. Якщо множина C^i містить тільки 1 елемент a_{c_1}, то для решти $n-i-1$ елементів, що не входять до поточної часткової перестановки π^i, умова (2.23) не виконується, тобто вони є більшими за π_i і, отже, більшими за a_{c_1}. Якщо множина C^i містить $p>1$ елементів, то кількість твірних елементів, що не входять до часткової перестановки π^i та не належать до C^i (тобто не задовольняють умові (2.23)), дорівнює $n-i-p$.

Якщо $n-i-p\geq m^i$, то умові (2.27) задовольняють всі елементи a_{c_k} множини C^i, тому що для кожного a_{c_k} завжди знайдеться $n-i-p$ твірних елементів, що є більшими за a_{c_k} і не входять до π^i.

Якщо $n-i-p<m^i$, то умові (2.27) задовольняють не всі елементи множини C^i, а тільки ті, для яких кількість твірних елементів, що не входять до π^i та є більшими за a_{c_k}, дорівнює m^i. Таким образом, необхідна наявність m^i елементів, що є більшими за a_{c_k}, а позі множиною їх тільки $n-i-p$, що значить необхідність існування ще $m^i-(n-i-p)$ елементів множини C^i, що є більшими за a_{c_k}. Звідси умова (2.27) виконується для перших $p-(m^i-(n-i-p))=n-i-m^i$ елементів множини C^i.

Для того, щоб твірний елемент задовольняв умові (2.25), він може бути як меншим останнього елементу π_i поточної часткової перестановки, так і більшим за нього. Відповідно, всі твірні елементи задовольняють умові (2.25).

Для випадку, коли наступна позиція $i+1$ не визначається частково заданою сигнатурою перестановки, ніякі додаткові умови не перевіряються.

Було показано, що на кожному рівні $i\in J_{n-1}^0$ множина F^i завжди містить щонайменше 1 елемент, що значить відсутність тупиків в рекурсивному дереві (backtracking tree), що будується в процесі роботи алгоритму генерації перестановок з частково заданою сигнатурою, і свідчить про справедливість теореми.

Твердження. Алгоритм *GenBase* для перестановок з частково заданою сигнатурою генерує їх в лексикографічному порядку.

Доведення. Як зазначалося вище, елементи кожного кортежу F^i є строго впорядкованими за зростанням: $f_1^i<f_2^i<...<f_{k^i}^i$, $f_j^i\in F^i$, $j\in J_{k^i}$, $k^i=Card\,F^i$. Алгоритм *GenBase* додає в поточну часткову перестановку π^i елементи $f_j^i\in F^i$, де j послідовно приймає значення $1,2,...,k^i$, і рекурсивно викликає сам себе з параметром $\pi^{i+1}, \pi_{i+1}=f_j^i\in F^i$. Це означає, що рекурсивні виклики *GenBase* відбуваються спочатку для елементу $f_1^i<f_2^i<...<f_{k^i}^i$, потім для

наступного елементу $f_2^i < ... < f_{k^i}^i$ і так далі в порядку зростання елементів $f_j^i \in F^i$, що і значить лексикографічний порядок.

2.4.5 Окремі випадки перестановок з частково заданою сигнатурою

Якщо $r(\pi)$ не визначено для жодної позиції, це значить, що відношення сусідніх елементів в перестановці неважливе, і перестановки з частково заданою сигнатурою зводяться до множини P_n всіх перестановок з n елементів.

Якщо в частково заданій сигнатурі $r(\pi)$ визначено всі позиції $i \in \{1,2,...,n-1\}$, перестановки з частково заданою сигнатурою вироджуються в перестановки з повністю заданою сигнатурою $\rho(\pi)$, або в перестановки з заданими підйомами та спусками [134]. Окремими випадками останнього класу перестановок є унімодальні перестановки [140] при $\rho(\pi) = (>,>,...,>,<,...,<)$ або $\rho(\pi) = (<,<,...,<,>,...,>)$ (тобто коли спочатку йдуть тільки спуски, а потім – тільки підйоми або навпаки), а також альтернативні перестановки [138] при $\rho(\pi) = (>,<,>,<,...)$ або $\rho(\pi) = (<,>,<,>,...)$ (тобто коли спуски та підйоми чергуються).

2.5 Висновки по другому розділу

У розділі запропоновано стратегію та розроблений в її рамках узагальнений метод генерації комбінаторних множин, що є базовим результатом, на використанні якого грунтуються усі подальші розроблені методи комбінаторної генерації, а також методи розв'язання задач комбінаторної оптимізації. Запропоновано алгоритмічну реалізацію розробленого узагальненого методу – алгоритм *GenBase*.

Розроблено методи та алгоритми повної та випадкової (часткової) генерації $k-$множин. Перевагою запропонованих методів є можливість отримання проміжних результатів генерації, тобто множин P^i на рівнях $i<k$. Досить важко визначити явну залежність часу виконання алгоритму генерації від вхідних даних, тому що базові множини можуть належати до різних класів комбінаторних множин, які мають різні властивості і різну залежність кількості елементів від вхідних даних. Однак для характерних випадків, таких як композиція перестановок, така залежність була отримана в явному вигляді.

Запропоновано спеціальний вид перестановок – перестановки з частково заданою сигнатурою, що є узагальненням перестановок з заданими підйомами та спусками. Оцінено кількість таких перестановок, побудовано метод їх генерації, що базується на узагальненому методі, та алгоритм генерації, що використовує алгоритм *GenBase*.

Результати розділу опубліковано в роботах [46], [48]–[52].

3 МАТЕМАТИЧНІ МОДЕЛІ ЗАДАЧ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ І МЕТОДИ ЇХ РОЗВ'ЯЗАННЯ

В даному розділі розроблено математичні моделі задач перевезення та обробки вантажів та розвинуто методи їх розв'язання за рахунок використання узагальненого методу комбінаторної генерації та врахування додаткових властивостей задач. Використовується стратегія розв'язання задач комбінаторної оптимізації, що полягає в описі областей допустимих розв'язків за допомогою некласичних комбінаторних конфігурацій та застосуванні методів комбінаторної генерації, описаних в розділі 2, для розв'язання таких задач.

3.1 Задача вивозу і доставки (PDP)

У даному пункті розглядається задача Pickup and Delivery Problem в постановці, що враховує додаткові властивості порівняно з існуючими дослідженнями: послідовність завантаження тривимірних контейнерів та порядок відвідування пунктів вивозу і доставки, виключаючи можливість блокування одних контейнерів іншими під час розвантаження та забезпечуючи стійкість завантаження.

Пропонується математична модель і стратегія розв'язання досліджуваної задачі, що використовує математичний апарат комбінаторних конфігурацій замість булевих змінних, що традиційно застосовуються для опису даного класу задач, та базується на використанні узагальненого методу комбінаторної генерації. Це дозволяє знизити розмірність, урахувати додаткові властивості та підвищити ефективність розв`язання задачі. Область допустимих розв'язків задачі описується за допомогою k-множини – композиції перестановок. Запропоновано точний та евристичний метод розв'язання досліджуваної задачі.

3.1.1 Постановка задачі

Розглядається класична задача PDP, схему якої представлено на рис. 3.1. Задано множину пар пунктів завантаження (pickup, квадрати на рис. 3.1) та доставки (delivery, круги на рис. 3.1). В кожному пункті pickup знаходиться вантаж (контейнер) з заданими розмірами та масою, який треба доставити у відповідний пункт delivery (кожному пункту pickup відповідає єдиний пункт delivery). Транспортні засоби, що стартують з депо (трикутник на рис. 3.1), повиннні об'їхати всі пункти pickup, вивезти з них контейнери та доставити їх у відповідні пункти delivery, не перетинаючись на своїх шляхах. Розв'язанням задачі є набір оптимальних (з точки зору цільової функції) маршрутів транспортних засобів, що дозволяють дозволяють доставити всі вантажі та відподвідають заданим обмеженням. Приклад такого маршруту для випадку одного транспортного засобу наведено в правій частині рис. 3.1.

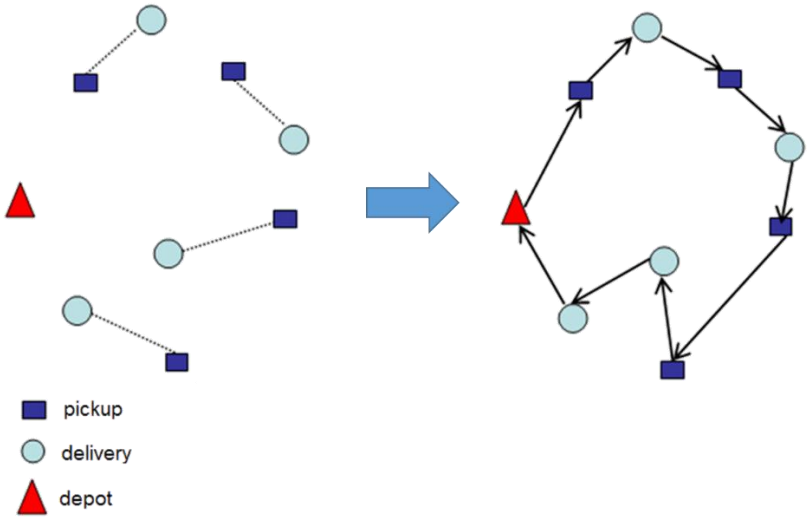


Рисунок 3.1. Схема задачі Pickup and Delivery Problem (PDP)

Розглядається симетричний випадок цієї задачі [144], коли кожна дуга (i, j) дорівнює дузі (j, i), і може бути замінена одним ребром. Нехай задано n пар пунктів вивозу (pickup) та доставки (delivery), тобто необхідно обслужити n клієнтів. В наявності є v ідентичних транспортних засобів, кожен з яких має вантажопідйомність Q та вантажний відсік у формі паралелепіпеду, що визначається шириною W, висотою H та довжиною L. Кожен клієнт i, $i \in J_n$,

$J_n = \{1, 2, ..., n\}$, потребує завантаження (pickup) або доставки (delivery) одного вантажу (контейнера) також у формі паралелепіпеду з шириною w_i, висотою h_i і довжиною l_i та загальною вагою q_i (пункти pickup асоціюються з додатнім значенням q_i, пункти delivery – з від'ємним значенням).

Розглядаються такі обмеження на завантаження контейнерів в транспортний засіб (далі – тривимірні обмеження завантаження, 3D loading constrains).

TD1. Ніякий вантаж (контейнер) не виходить за межі вантажного відсіку транспортного засобу.

TD2. Ніяка одиниця вантажу не перетинається с іншими.

TD3. Одиниці вантажу можуть бути поміщені у середину транспортного засобу лише ортогонально, при цьому їх грані повинні бути паралельні граням області розміщення; при цьому допускається можливість повороту кожної одиниці вантажу на $90°$ у горизонтальній площині довжини - ширини.

TD4. Обмеження стійкості: кожен вантаж (контейнер) повинна стояти або безпосередньо на підлозі вантажного відсіку, або зверху іншого контейнера. Проте в останньому випадку контейнер зверху повинен повністю підтримуватися контейнером знизу, тобто не допускається «висіння у повітрі» ніякої частини одиниці вантажу.

TD5. Ще однією важливою вимогою є забезпечення того, щоб контейнери у пункті pickup завантажувалися таким чином, щоб вони могли бути легко вивантажені у відповідному пункті delivery. При відвідуванні пункта delivery відповідна одиниця вантажу не повинна знаходитись під іншою одиницею вантажу або бути заблокованою вантажами для тих пунктів delivery, які повинні бути відвідані пізніше. Одиниця вантажу вважається *заблокованою спереду*, якщо вона перетнеться з будь-якою іншою одиницею вантажу під час свого переміщення уздовж осі *Y* (що відповідає довжині вантажного відсіку *L*) до задніх дверей у момент вивантаження в пункті delivery (рис. 3.2, а); *заблокованою зверху* – якщо на ній знаходиться інша

одиниця вантажу (рис. 3.2, б).

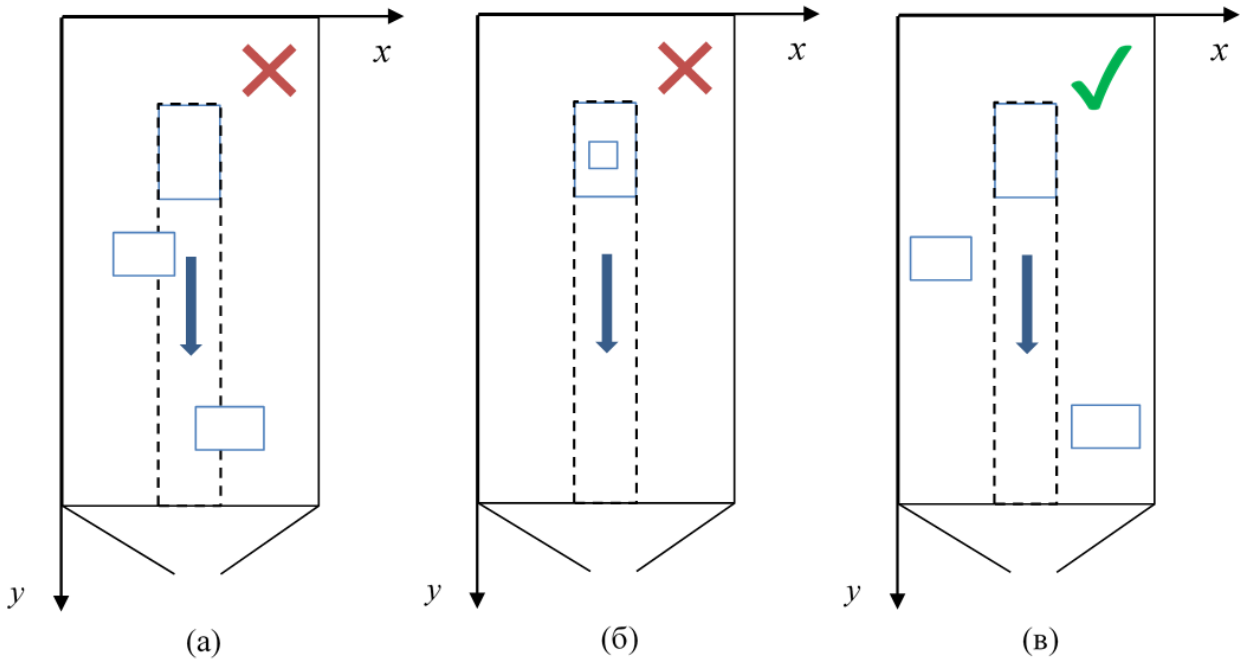


Рисунок 3.2. Приклади розташування контейнерів у вантажному відсіку, при яких контейнер блокується спереду (а), блокується зверху (б) та не блокується (в) іншими контейнерами при вивантаженні

Позначимо задачу в даній постановці як задачу PDP з тривимірними обмеженнями завантаження, запобіганням блокуванню та перевіркою стійкості (PDP with 3D loading constraints, blocking prevention and stability check, PDPBS).

Розв'язанням задачі є оптимальна послідовність з $\mu \leq \nu$ маршрутів ідентичних транспортних засобів (допускається, що не всі транспортні засоби можуть приймати участь у обслуговуванні клієнтів) така, щоб:

BS1: пункти pickup відвідувались раніше відповідних пунктів delivery;

BS2: загальна вага всіх вантажів, що перевозяться кожним транспортним засобом в кожен момент часу, не перевищувала б його вантажопідйомність;

BS3: усі одиниці вантажу для всіх клієнтів були б завантажені у відповідний транспортний засіб таким чином, щоб задовольняти тривимірні

обмеження завантаження TD1-TD5.

Таким чином, задача PDPBS полягає в знаходженні оптимальної послідовності маршрутів множини ідентичних транспортних засобів, що дозволяють здійснити розвезення вантажів (заданих розмірів та маси) з усіх пунктів pickup до відповідних пунктів delivery та задовольняють обмеження BS1-BS3.

У якості цільової функції задачі PDPBS може виступати сумарна вартість розвезення, час розвезення всіх вантажів, кількість задіяних транспортних засобів тощо.

3.1.2 Математична модель задачі PDPBS

Введемо позначення.

n ... – кількість клієнтів (пар pickup-delivery);

P ... – множина пунктів вивозу (pickup), $P = \{1, 2, ..., n\}$;

D ... – множина пунктів доставки (delivery), $D = \{n+1, n+2, ..., 2n\}$;

q_i – маса контейнера, що завантажується в транспортний засіб або вивантажується з нього в пункті i; пункти pickup асоціюються з додатнім значенням q_i, а пункти delivery – з від'ємним її значенням;

w_i, h_i, l_i – відповідно ширина, висота і довжина завантажуваного (розвантажуваного) контейнера в пункті $i \in J_{2n}$;

v – кількість транспортних засобів;

Q – вантажопідйомність кожного транспортного засобу;

W, H, L – відповідно ширина, висота і довжина вантажного відсіку кожного транспортного засобу;

$C = \{(p_1, d_1), (p_2, d_2), ..., (p_n, d_n)\}$ – множина клієнтів (пар pickup-delivery);

$(p_i, d_i) \in C$ – пара відповідних пунктів pickup та delivery, $p_i \in P$, $d_i \in D$,

$d_i = p_i + n,\ i \in J_n$;

μ – кількість транспортних засобів, що приймають участь у обслуговуванні пар pickup-delivery, $\mu \le \nu$;

$C_1, C_2, ..., C_\mu$ – розбиття множини C, $C = \bigcup_{j=1}^{\mu} C_j$, $C_i \cap C_j = \varnothing$, $i \in J_\mu$, $j \in J_\mu$; існує взаємно однозначна відповідність між кожною підмножиною пар пунктів pickup-delivery C_j та транспортним засобом j, який обслуговує ці пункти, $j \in J_\mu$;

$c(i, j)$ – вартість переміщення транспортного засобу між пунктами *i* та *j*;

$\mathrm{B}_j = \{i_1^j, i_2^j, ..., i_{2n_j}^j\}$ – множина всіх пунктів, включених до C_j, які повинен об'їхати транспортний засіб j, $n_j = Card\, C_j$, $j \in J_\mu$, $\sum_{j=1}^{\mu} n_j = n$;

$P(\mathrm{B}_j)$ – множина перестановок, породжених елементами множини B_j, що описує всі можливі маршрути транспортного засобу j, тобто всі можливі послідовності відвідування пунктів B_j, що обслуговуються транспортним засобом.

Позначимо через $u_0^j = (0,0,0)$ полюс (точку початку власної системи координат) вантажного відсіку транспортного засобу $j \in J_\mu$, що знаходиться у вершині параллелепіпеду, що відповідає вантажного відсіку (див. рис. 3.3). Початок власної системи координат кожного контейнера також знаходиться у

вершині відповідного параллелепіпеду (рис. 3.4).

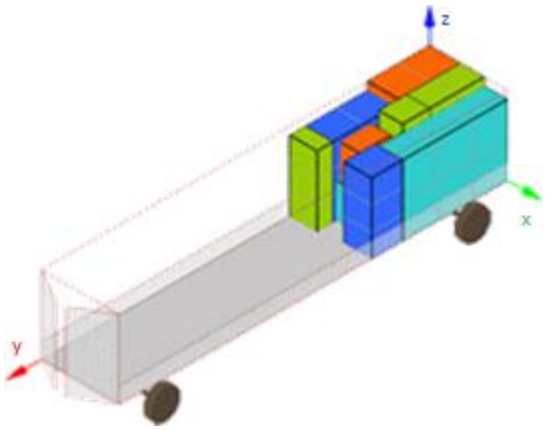


Рисунок 3.3. Схема вантажного відсіку транспортного засобу

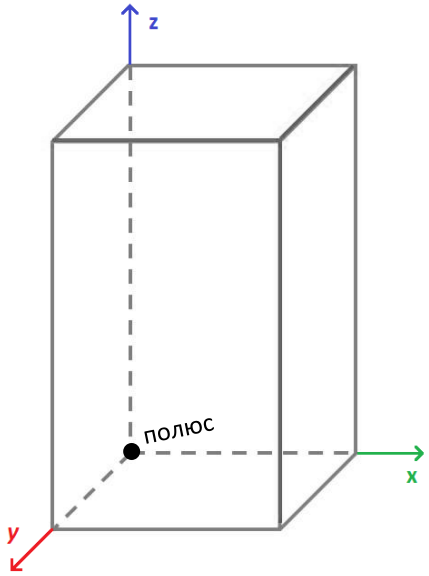


Рисунок 3.4. Система координат окремого контейнера

Розміщення контейнера $i \in \mathrm{B}_j$ (номер контейнера відповідає номеру пункта pickup або delivery) в транспортному засобі $j \in J_\mu$ задається вектором трансляції $u_i^j = (x_i^j, y_i^j, z_i^j)$ початку власної системи координат контейнера відносно початку власної системи координат вантажного відсіку, а також розміщенням контейнера, що визначається його поворотом на кут $\pi / 2$ в площині ширини-довжини $v_i^j = (lw_i, h_i), lw_i \in \{(l_i, w_i), (w_i, l_i)\}$ або

$\nu_i^j = (\nu_{1i}^j, \nu_{2i}^j, \nu_{3i}^j)$, $i \in \mathrm{B}_j$, $j \in J_\mu$.

Введемо наступні позначення:

$.U = (U^1, U^2, ..., U^\mu) \in \mathbb{R}^{3n}$. – вектори трансляції (координати полюсів) усіх контейнерів $i \in \mathrm{B}_j$ в усіх транспортних засобах $j \in J_\mu$: $U^j = (u_1^j, u_2^j, ..., u_{2n_j}^j)$, $u_i^j = (x_i^j, y_i^j, z_i^j)$, $i \in \mathrm{B}_j$, $j \in J_\mu$;

$V = (V^1, V^2, ..., V^\mu) \in \mathbb{R}^{3n}$ – лінійні розміри контейнерів в порядку, який визначає їх орієнтацію (перестановка лінійних розмірів відповідає повороту паралелепіпеда на кут $\pi / 2$), $V^j = (v_1^j, v_2^j, ..., v_{2n_j}^j)$, $\nu_i^j = (lw_i, h_i), lw_i \in \{(l_i, w_i), (w_i, l_i)\}$ або $\nu_i^j = (\nu_{1i}^j, \nu_{2i}^j, \nu_{3i}^j)$, $i \in \mathrm{B}_j$, $j \in J_\mu$;

$v_0^j = (W, L, H)$ – розміри вантажного відсіку кожного транспортного засобу $j \in J_\mu$;

$\pi^j = (i_1^j, i_2^j, ..., i_{2n_j}^j) \in P(\mathrm{B}_j)$ – маршрут кожного транспортного засобу $j \in J_\mu$.

Побудуємо відображення векторів π^j , $j \in J_\mu$ у евклідів простір [9]:

$$
\begin{aligned}
& f : P(\mathrm{B}^j) \to R^{2n_j}, \forall \pi^j = (i_1^j, i_2^j, ..., i_{2n_j}^j) \in P(\mathrm{B}^j), \\
& e^j = f(\pi^j) = (e_1^j, e_2^j, ..., e_{2n_j}^j) \in E^j \subset R^{2n_j}, \\
& e_k^j = i_k^j, k \in J_{2n_j}, \\
& j \in J_\mu
\end{aligned}
\tag{3.1}
$$

У результаті відображення f кожному елементу множини $P(\mathrm{B}^j)$ буде поставлена у відповідність точка множини E^j, що належить евклідовому простору R^{2n_j}.

Розв’язком задачі PDPBS є вектор, що містить маршрути кожного

транспортного засобу, якому відповідає мінімальна сумарна вартість перевезень

$$E = (e^1, e^2, ..., e^\mu) \in \mathbb{R}^{2n}. \quad (3.2)$$

Наведемо математичний опис обмежень задачі.

Обмеження послідовності відвідування точок pickup та delivery (BS1) описуються таким чином. Для кожного пункту доставки (delivery) $e_r^j \in D$ маршруту кожного транспортного засобу $j \in J_\mu$ відповідний пункт pickup $e_t^j \in P$ має відвідуватися раніше ($t < r$):

$$e_r^j \in D \Rightarrow \exists t < r : e_t^j \in P, e_t^j + n = e_r^j,\ r \in J_{2n_j},\ j \in J_\mu. \quad (3.3)$$

Обмеження вантажопідйомності (BS2) формулюється так: загальна маса всіх вантажів, що перевозяться транспортним засобом $j \in J_\mu$ на момент досягнення кожного пункту $s \in J_{2n_j}$ маршруту e^j

$$Q(j,s) = \sum_{k=1}^{s} q_{e_k^j}, \quad (3.4)$$

не має перевищувати його вантажопідйомність

$$Q(j,s) \le Q,\ s \in J_{2n_j}. \quad (3.5)$$

Для математичного моделювання тривимірних обмежень завантаження (3D loading constraints) використаємо математичний апарат Ф-функцій, що запропонований в роботах Ю.Г.Стояна та М.І. Гіля для формального опису взаємного положення *n*-вимірних об’єктів [205]–[207]. Зокрема, в задачі упаковки [206] Ф-функції дозволяють формально описати умови взаємного неперетину двох паралелепіпедів та умови невиходу паралелепіпеда за межі

області розміщення, що також має форму параллелепіпеда.

Для опису *тривимірних обмежень завантаження* TD1-TD3 використаємо дві функції, які при фіксації значень дискретних змінних v_i^j стають Ф-функціями:

– $\Phi_{im}^j(u_i^j, u_m^j, v_i^j, v_m^j)$ – для перевірки того, що контейнер i (що визначається координатами полюса u_i^j і орієнтацією v_i^j) не перетинається з контейнером m (що визначається координатами полюса u_m^j і орієнтацією v_m^j)

$$\begin{aligned}\Phi_{im}^j(u_i^j, u_m^j, v_i^j, v_m^j) = \max\{&x_m^j - x_i^j - \nu_{1i}^j, -x_m^j + x_i^j - \nu_{1m}^j,\\ &y_m^j - y_i^j - \nu_{2i}^j, -y_m^j + y_i^j - \nu_{2m}^j, z_m^j - z_i^j - \nu_{3i}^j, -z_m^j + z_i^j - \nu_{3m}^j\},\end{aligned} \tag{3.6}$$

– $\Phi_{0m}^j(u_0^j, u_m^j, v_m^j)$ – для перевірки можливості розташування контейнера m в вантажному відсіку D_j (що визначається W, H, L – шириною, висотою і довжиною відповідно).

$$\begin{aligned}\Phi_{0m}^j(u_0^j, u_m^j, v_m^j) = \min\{&x_m^j - x_0^j, -x_m^j + x_0^j + W - \nu_{1m}^j,\\ &y_m^j - y_0^j, -y_m^j + y_0^j + L - \nu_{2m}^j, z_m^j - z_0^j, -z_m^j + z_0^j + H - \nu_{3m}^j\}.\end{aligned} \tag{3.7}$$

Якщо

$$\Phi_{im}^j(u_i^j, u_m^j, v_i^j, v_m^j) \geq 0 \tag{3.8}$$

для всіх $i, m \in J_n, i < m$, то будь-яка пара контейнерів у вантажному відсіку транспортного засобу j не перетинається.

Якщо

$$\Phi_{0m}^j(u_0^j, u_m^j, v_m^j) \geq 0 \tag{3.9}$$

для всіх $m \in J_n$, то кожен контейнер може бути розміщений у вантажному відсіку.

Тому розміщення контейнерів в кожному транспортному засобі повинно

бути проведене таким чином, щоб виконувалися умови (3.8)-(3.9).

Обмеження стійкості (контейнери, що завантажені зверху, повинні стояти стабільно, TD4) можна описати таким чином. Для кожного транспортного засобу $j \in J_\mu$ повинна виконуватись умова: якщо контейнер $k \in \mathrm{B}^j$ не лежить безпосередньо на підлозі вантажного відсіку ($z_k^j > 0$), то повинен існувати контейнер $p \in \mathrm{B}^j$, такий, що дно контейнера *k* повністю знаходиться на верхній площині контейнера *p* ($z_k^j = z_p^j + h_p^j$):

$$\begin{aligned} \forall k \in \mathrm{B}^j : z_k^j > 0 \Rightarrow \exists p \in \mathrm{B}^j : & z_k^j = z_p^j + h_p^j, \\ & x_k^j < x_p^j, y_k^j < y_p^j, \\ & x_k^j + v_{1k}^j < x_p^j + v_{1p}^j, y_k^j + v_{2k}^j < y_p^j + v_{2p}^j, \\ j \in J_\mu. & \end{aligned} \tag{3.10}$$

Обмеження неблокування одними контейнерами інших при вивантаженні (TD5) можна описати таким чином. Для кожного пункту доставки (delivery) $e_r^j \in D,\ r \in J_{2n_j},\ j \in J_\mu$ в маршруті e^j можуть існувати пункти pickup $e_t^j \in P$, що відвідуються раніше пункта *r* ($t < r$), а відповідні їм пункти delivery $d : e_t^j + n = e_d^j$ відвідуються пізніше пункта r ($d > r$). Позначимо множину відповідних контейнерів як $t \in T$. Множина *T* позначає всі інші контейнери, які на поточний момент знаходяться у транспортному засобі.

Обмеження неблокування спереду: контейнери $t \in T$ не повинні стояти на шляху вивантаження контейнера r, тобто не перетинатися з уявним паралелепіпедом ry', полюс якого співпадає з полюсом u_r^j контейнера *r*, а лінійні розміри визначаються як $v'^j_{ry} = (v_{1r}^j, L - u_{2r}^j, v_{3r}^j)$. Цей уявний паралелепіпед включає до себе контейнер *r* та шлях, увздовж якого контейнер переміщується при вивантаженні. Йому відповідає пунктирна лінія на рис. 3.2.

Умова такого неперетину описується нерівністю

$$\Phi_{rt}^{j}(u_t^j, u_r^j, v_t^j, v'^{j}_{ry}) \geq 0.$$

Обмеження неблокування зверху: жоден з контейнерів $t \in T$ не повинен стояти на контейнері *r*, тобто не перетинатися з уявним паралелепіпедом rz', полюс якого співпадає з полюсом u_r^j контейнера *r*, а лінійні розміри визначаються як $v'^{j}_{rz} = (v_{1r}^j, v_{2r}^j, H - u_{3r}^j)$. Цей уявний паралелепіпед включає до себе контейнер *r* та всю область над ним до «стелі» вантажного відсіку. Умова такого неперетину описується нерівністю

$$\Phi_{rt}^{j}(u_t^j, u_r^j, v_t^j, v'^{j}_{rz}) \geq 0.$$

Таким чином, обмеження неблокування формально описуються як

$$\begin{aligned} & e_r^j \in D: \\ & \quad \exists T = \{t < r : e_t^j \in P, \exists d > r : e_t^j + n = e_d^j\} \neq \varnothing \\ & \quad \Rightarrow \Phi_{rt}^{j}(u_t^j, u_r^j, v_t^j, v'^{j}_{ry}) \geq 0, v'^{j}_{ry} = (v_{1r}^j, L - u_{2r}^j, v_{3r}^j), \\ & \qquad \Phi_{rt}^{j}(u_t^j, u_r^j, v_t^j, v'^{j}_{rz}) \geq 0, v'^{j}_{rz} = (v_{1r}^j, v_{2r}^j, H - u_{3r}^j), \\ & \qquad \forall t \in T, \\ & r \in J_{2n_j}, j \in J_{\mu}. \end{aligned} \tag{3.11}$$

Приклад. Нехай *n*=3, один транспортний засіб ($\mu = 1$) і його маршрут

$$e^1 = (3\,2\,5\,1\,4\,6)$$

(дугами помічено зв'язок між пунктами pickup та delivery, наприклад, пункту pickup $e_1^1 = 3$ відповідає пункт delivery $e_6^1 = 6$).

Розглянемо умову неблокування для пункта delivery $e_5^1 = 4$ ($r = 5$). В маршруті є лише один пункт pickup $e_1^1 = 3$ (тобто $T = \{1\}$), що відвідується раніше, ніж пункт $r = 5$ ($1 < 3$), а відповідний йому пункт delivery

$d = 6 : e_1^1 + 3 = e_6^1 = 6$ відвідується пізніше, ніж пункт $r = 5 (6 > 5)$. Тоді умова неблокування потребує, щоб контейнер 3 не блокував (спереду та зверху) контейнер 4.

Варто відзначити, що контейнер 2 був завантажений і вивантажений перед пунктом $r = 5$, тому в умові неблокування не перевіряється пункт $e_2^1 = 2$ та відповідний йому контейнер 2 (бо він вже відсутній у транспортному засобі).

Цільовою функцією задачі є сумарна вартість всіх маршрутів

$$\Psi(E,U,V) = \sum_{j=1}^{\mu} [c(0, e_1^j) + \sum_{k=1}^{2n_j - 1} c(e_k^j, e_{k+1}^j) + c(e_{2n_j}^j, 2n+1)], \qquad (3.12)$$

де $c(0, e_1^j)$ – відстань від депо (що описується за допомогою фіктивного пункту 0) до першого відвіданого пункту;

$c(e_{2n_j}^j, 2n+1)$ – відстань від останнього відвіданого пункту до депо (фіктивні пункти 0 і $2n+1$ відповідають депо);

e_k^j – k-тий пункт маршруту транспортного засобу j.

Математична модель задачі PDPBS наведена нижче.

$$\min_{(E,U,V) \in \mathbb{W}} \Psi(E,U,V)$$
$$\mathbb{W} = \{(E,U,V) \in \mathbb{R}^{8n}\}, \qquad (3.13)$$

де (E,U,V) задовольняють обмеження (3.5), (3.8)-(3.11).

3.1.3 Стратегія розв’язання задачі PDPBS

Пропонується стратегія розв’язання задачі, що містить 2 етапи.

3.1.3.1 Перший етап – кластеризація

На першому етапі множина пар pickup-delivery C розбивається на

підмножини або кластери C_j, $j \in J_\mu$. Кожен кластер C_j містить пари пунктів pickup та delivery (p_i, d_i), що повинні обслуговуватися транспортним засобом j.

Для розв'язання задачі кластеризації було обрано метод *k*-середніх (*k*-means) [225]–[227], що дозволяє розбити задану множину точок на задану кількість кластерів, що для даної задачі дорівнює μ. Метод полягає в повторенні наступних ітераційних кроків:

1. Обирається число кластерів μ. Із заданої множини точок випадковим чином обираються μ точок, які служать початковими центрами кластерів.
2. Для кожної точки обчислюється найближчий до неї центр кластера. Таким чином, утворюються початкові кластери.
3. Обчислюються центроїди – центри ваги кластерів. Кожен центр ваги – це точка, координати якої є середніми значеннями координат точок, що входять до кластера. Центр кластера зміщується в його центр ваги.
4. Кроки 3 та 4 ітеративно повторюються. Метод зупиняється тоді, коли цетроїди кластерів перестають змінюватися від ітерації до ітерації.

Перевагою методу є швидкість і простота його реалізації. До недоліків можна віднести невизначеність вибору початкових центрів кластерів.

Класичний метод *k*-середніх використовує одиничні точки, але в даній постановці задачі необхідно розбити на кластери не поодинокі пункти, а пари пунктів (p_i, d_i). Тому кожній парі (p_i, d_i) поставимо у відповідність одну точку k_i, координати якої є середнім арифметичним між координатами пунктів p_i і d_i

$$k_i.x = \frac{p_i.x + d_i.x}{2}, \; k_i.y = \frac{p_i.y + d_i.y}{2},$$

де $.x,.y$ – координати відповідного пункту.

У результаті роботи методу формується μ кластерів або множин пар пунктів pickup-delivery C_j, які обслуговуються транспортними засобами $j \in J_\mu$.

3.1.3.2 Другий етап – побудова маршруту.

На другому етапі розглядається кластер C_j, який обслуговується транспортним засобом j. Необхідно побудувати оптимальний маршрут для цього транспортного засобу, тобто розв'язати задачу PDPBS для випадку одного транспортного засобу.

Множина E^j (що відповідає множині перестановок $P(\mathrm{B}_j)$) описує маршрут транспортного засобу. Також кожен елемент цієї множини $e^j \in E^j$ (що відповідає перестановці $\pi^j \in P(\mathrm{B}_j)$) визначає порядок відвідування пунктів pickup та delivery транспортним засобом j та, відповідно, порядок завантаження контейнерів в транспортний засіб (що відбувається у пунктах pickup) і їх вивантаження (що відбувається у пунктах delivery).

Маршрут e^j має задовольняти всім описаним вище тривимірним обмеженням завантаження. Щоб враховувати повороти контейнерів у горизонтальній площині довжини-ширини, кожен елемент в e^j замінюється вектором $lw_i \in \{(l_i, w_i), (w_i, l_i)\}, i \in \mathrm{B}_j$.

Така комбінаторна структура описується за допомогою *k*-множини – композиції перестановок [44].

Для побудови оптимального маршруту для транспортного засобу j необхідно вибрати оптимальну (згідно (3.12)) перестановку його пунктів B_j та орієнтацію кожного контейнера у горизонтальній площині довжини-ширини $lw_i \in \{(l_i, w_i), (w_i, l_i)\}$.

Таким чином, описана задача зводиться до задачі комбінаторної

оптимізації на множині композиції перестановок.

3.1.4 Метод розв'язання для другого етапу

В даному пункті описується розв’язання задачі PDP на другому етапі, тобто розв’язання задачі пошуку оптимального маршруту e^j для транспортного засобу j, $j \in J_\mu$.

3.1.4.1 Точне розв'язання

Для розв'язання задачі генерації маршрутів e^j використовується алгоритм *GenBase*.

Для зручності подальшого опису позначимо маршрут транспортного засобу j як $t = e^j$, а перші пункти *i* маршруту *t* назвемо частковим маршрутом $t^i = (t_1, t_2, ..., t_i)$.

В термінах даної задачі, на кожному рівні $i \in J^0_{2n-1}$, $J^0_{2n-1} = \{0,1...2n-1\}$ алгоритм *GenBase* розширює поточний частковий маршрут $t^i = (t_1, t_2, ..., t_i)$, додаючи наступний пункт t_{i+1} в кінець маршруту, отримуючи, таким чином, на наступному рівні $i+1$ новий частковий маршрут $t^{i+1} = (t_1, t_2, ..., t_{i+1})$. На рівні $i=2n$ алгоритм отримує повний маршрут $t^{2n} = t$ і додає його до результуючої множини *T*. Іншими словами, на кожній ітерації алгоритм додає новий пункт до поточного часткового маршруту до тих пір, поки не буде отримано повного маршруту.

Як відомо з описаного раніше принципу роботи алгоритму *GenBase*, елементи t_{i+1} повинні задовольняти деяким обмеженням, що зумовлені особливостями кожної конкретної комбінаторної множини. Кортеж всіх твірних елементів (в термінах даної задачі – пунктів pickup та delivery), що задовольняють цим обмеженням, на кожному рівні $i \in J^0_{2n-1}$ позначається як

$F^i = (f_1, f_2, ..., f_k)$. Таким чином, для кожного $j \in J_k, J_k = \{1,2...k\}$ алгоритм додає новий пункт $t_{i+1} = f_j$ до поточного часткового маршруту $t^i = (t_1, t_2, ..., t_i)$ і викликає себе рекурсивно, передаючи у якості нового часткового маршруту $t^{i+1} = (t_1, t_2, ..., f_j)$.

Відповідно до описаної постановки задачі Pickup and Delivery Problem, у кортеж F^i потрапляють пункти, що відповідають наступним обмеженням:

1) *обмеження унікальності*: пункти t^i є унікальними

$$t_{i+1} \neq t_z, z = 1..i;$$

2) *обмеження порядку слідування пунктів pickup-delivery*: кожен пункт pickup має бути відвіданий раніше, ніж відповідний йому пункт delivery. Це означає, що пункт delivery може бути доданий до маршруту t^i тільки тоді, коли маршрут вже містить відповідний пункт pickup:

$$t_{i+1} > n \Rightarrow \exists z : (t_{i+1} - n) = t_z$$

Наприклад, коли n = 4 (задано чотири пари pickup-delivery), delivery-пункт 5 може бути доданий до маршруту тільки якщо відповідний їй pickup-пункт 1 вже був включений до цього маршруту;

3) *обмеження вантажопідйомності* (3.5) на максимальну вантажопідйомність транспортного засобу;

4) тривимірн*і обмеження завантаження:* якщо t_{i+1} – pickup, то новий контейнер буде поміщений у транспортний засіб. Це вимагає перевірки тривимірних обмежень завантаження. Для перевірки цих обмежень використовується дещо модифікований алгоритм, описаний в роботі [206]. Далі будемо позначати цей алгоритм як *CheckLoadingConstrains*. Треба зазначити, що алгоритм [206] при необхідності може повертати кожен елемент в горизонтальній площині, таким чином визначаючи вектори $lw_i,\ i \in \mathrm{B}_j$.

Процедура *GetF_PDP*, що використовується для формування кортежу

F^i та передається в алгоритм *GenBase*, наведена нижче.

```
def GetF_PDP (t^i , p={B_j, n, Q})
  result = ( )
  for t in B_j
    not_in_path = t ≠ t_z, z = 1..i
    is_pickup = t > n
    precedence_is_correct = (not is_pickup and ∃z : (t − n) = t_z)

    weight_constaints = Q(j, s) ≤ Q , s ∈ J_{2n_j}
   loading_constrains = CheckLoadingConstrains(t^i , p)
   if not_in_path and precedence_is_correct and weight_constraints
    and loading_constrains:
      result.append(t)
  return result
```

Алгоритм *GenBase* у процесі своєї роботи будує рекурсивне дерево, де кожна вершина на рівнях $i < 2n\text{-}1$ є частковим маршрутом, а вершини на останньому рівні $i = 2n\text{-}1$ є повними маршрутами. Під час роботи алгоритму на рівнях $i < 2n\text{-}1$ розширюється кожен лист дерева, тобто в частковий маршрут додається кожна нова вершина з кортежу F^i.

Приклад. Продемонструємо роботу алгоритму при побудові маршрутів для $n = 2$ (пункти 1, 2 – pickup, а пункти 3,4 – delivery).

На початку, на рівні $i = 0$, F^0 складається тільки з пунктів pickup: $F^0 = (1,2)$. Кожен пункт pickup додається в кінець часткового (на даний момент – порожнього) маршруту $t^0 = (\ )$, створюючи нові часткові маршрути $t^1 = (1)$ та $t^1 = (2)$. Після цього *GenBase* викликається рекурсивно для кожної щойно збудованої вершини t^1.

На наступному рівні $i = 1$ для маршруту $t^1 = (1)$ можна додати або pickup-пункт 2, або delivery-пункт 3, відповідну pickup-пункту 1. Пункт 4 не може бути доданий, тому що в поточному маршруті $t^1 = (1)$ на даний момент немає відповідного pickup-пункту 2.

На рис. 3.5 наведено рекурсивне дерево, збудоване у процесі роботи алгоритму *GenBase*.

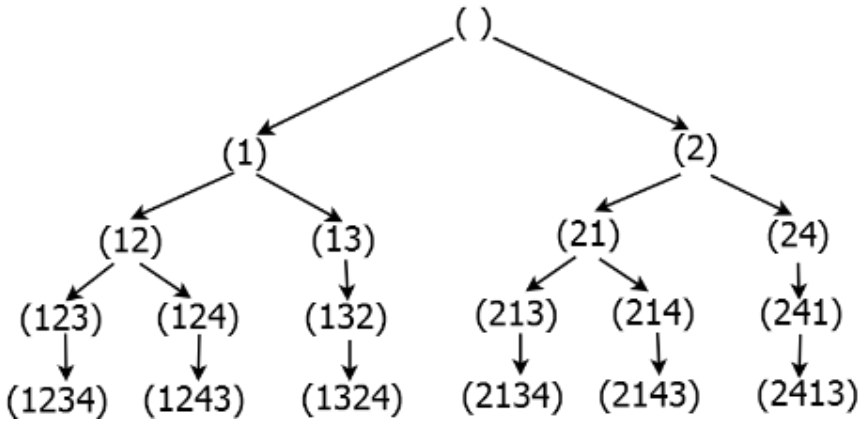


Рисунок 3.5. Рекурсивне дерево, побудоване у процесі роботи алгоритму *GenBase*

Коли всі маршрути згенеровані (рекурсивне дерево повністю збудоване), як остаточний розв’язок задачі обирається маршрут з кращим значенням цільової функції (3.12).

3.1.4.2 Евристичне розв’язання

Оскільки реальні задачі PDP містять значну кількість пар pickup-delivery, виникає необхідність у застосуванні евристик в методах розв’язання таких задач. В даній роботі було обрано евристику променевого пошуку (beam search) [228], що дозволяє згенерувати не всю область допустимих розв’язків, а деяку її підмножину, що містить «кращі» (за обраним критерієм) рішення.

Застосуємо евристику променевого пошуку в алгоритмі *GenBase*. В термінах даної задачі, евристика променевого пошуку працює на етапі розширення (expanding) рекурсивного дерева, тобто на етапі формування кортежу F^i. Застосування променевого пошуку означає включення до F^i не всіх пунктів, що задовольняють описаним вище обмеженням , а тільки деякої частини з них, що мають краще значення цільової функції. Інші пункти виключаються з подальшого розглядання.

Ілюстрацію принципу роботи евристики променевого пошуку [221] наведено на рис 3.6.

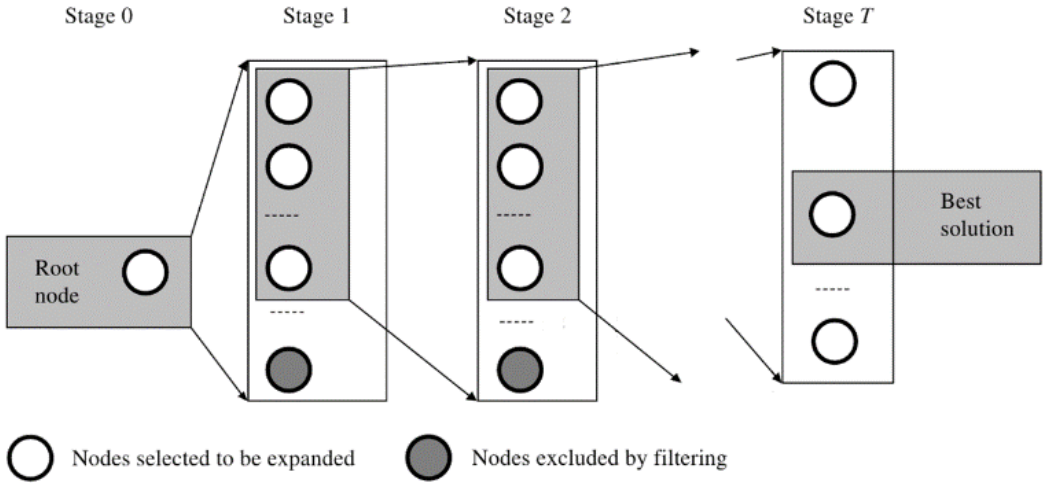


Рисунок 3.6. Ілюстрація принципу роботи променевого пошуку [221]

Було прийняте рішення дещо модифікувати евристику променевого пошуку. У класичному променевому пошуку ширина променю (beam width) є константою, тобто для подальшого розширення рекурсивного дерева завжди обирається фіксована кількість пунктів (що дорівнює ширині променю). Під час розв'язання даної задачі виявилося більш ефективним обирати не фіксовану кількість пунктів, а деякий відсоток з них. Назвемо цей відсоток відносною шириною променю (relative beam width) та позначимо як *RBW*. Мінімальне значення *RBW* – 0, максимальне – 100 (у цьому випадку евристичне розв'язання зводиться до точного розв'язання).

Таким чином, евристика променевого пошуку використовується в даній задачі таким чином.

H1. Пункти в F^i сортуються за зростанням відстані від:

• депо, якщо $i = 0$ (перший пункт маршруту повинна знаходитися якомога ближче до депо);

• останнього пункту t_i в поточному маршруті, якщо $i > 0$.

H2. Рекурсивні виклики алгоритму *GenBase* виконуються тільки для перших *RBW%* пунктів з кортежу F^i, де *RBW* є константою. Інші пункти виключаються з подальшого розглядання.

Також для прискорення роботи алгоритму введемо наступну евристику.

НЗ. Оскільки перевірка тривимірних обмежень є складною з точки зору обчислювальних ресурсів процедурою, вони перевіряються не кожен раз, коли додається pickup, а з певною ймовірністю *check_prob* $\in$ [0; 1] на всіх рівнях, крім останнього $i = 2n\text{-}1$. На останньому рівні $i = 2n\text{-}1$ тривимірні обмеження перевіряються завжди, щоб виключити можливість того, що некоректний маршрут стане остаточним розв'язанням.

Опишемо процедуру *GetF_PDP_heuristic*, що є модифікацією процедури *GetF_PDP* і реалізує описані вище евристики. Цільова функція передається як *target_func* у множині параметрів *p*.

Важливо відмітити, що, змінюючи значення *RBW*, можна балансувати між часом роботи алгоритму і точністю отриманих результатів.

```
def GetF_PDP_heuristic (t^i, p={B_j, n, Q, RBW, check_prob, target_func})
  result = ( )
  for t in B_j
    not_in_path = t ≠ t_z, z = 1..i
    is_pickup = t > n
    precedence_is_correct = (not is_pickup
and ∃z : (t − n) = t_z)

     weight_constaints = Q(j, s) ≤ Q, s ∈ J_{2n_j}
    if random() < check_prob
      loading_constrains = CheckLoadingConstrains(t^i, p)
    else
      loading_constrains = True

    if not_in_path and precedence_is_correct and weight_constraints
                    and loading_constrains:
      result.append(t)
  count = len(result) * RBW / 100
  nodes_to_be_expanded = sorted(result, key=target_func)
  nodes_to_be_expanded = nodes_to_be_expanded[count:]
  return nodes_to_be_expanded
```

Приклад. Покажемо, як описана евристика працює при $n = 4$ (пункти 1-4 є пунктами pickup, а пункти 5-8 – пунктами delivery). Приймемо *RBW* = 50%; це означає, що на всіх рівнях $i < 2n-1$ до кортежу F^i буде включено половину всіх пунктів, що задовольняють обмеженням BS1-BS3.

На початку, на рівні $i = 0$, F^0 включає всі пункти pickup: $F^0 = (1,2,3,4)$. Далі проводиться їх сортування за відстанню до депо. Нехай відсортований кортеж має вигляд (2,4,3,1). З нього для подальшої роботи обираються перші 4*50/100=2, тобто, в даному випадку, пункти 2 і 4. Пункти 1 і 3 будуть виключені з подальшого розгляду. Таким чином, $F^0 = (2,4)$

Такий самий принцип працює на наступних рівнях. Розглянемо роботу алгоритму на рівні i=1 для часткового маршруту $t^1 = (2)$. Було відібрано чотири пункти-кандидати: три інших pickup-пункти 1,3 і 4, а також delivery-пункт 6, що відповідає pickup-пункту 2 ($6 = 2 + n$, $n = 4$). Отже, маємо кортеж кандидатів $(1,3,4,6)$. Припустимо, що пункти 1 і 6 є найближчими до пункти 2, тому розширимо їх, виключивши, таким чином, з подальшого розглядання пункти 3 і 4. Отже, $F^1 = (1,6)$, тому на наступному рівні i=2 вхідними поточними частковими маршрутами будуть $t^2 = (21)$ та $t^2 = (26)$ відповідно.

На рис. 3.7 представлено фрагмент рекурсивного дерева, збудованого у процесі роботи алгоритму *GenBase* для $n = 4$. Пункти, що були виключені з подальшого розглядання (excluded nodes), позначено пунктиром.

Зауваження 3.1. Застосування евристики *НЗ* теоретично може призвести до створення маршруту t^i, який не задовольняє тривимірним обмеженням на передостанньому рівні $i = 2n-2$ (оскільки *НЗ* означає перевірку тривимірних обмежень не завжди, а лише з деякою ймовірністю *check_prob*). При цьому на останньому рівні $i = 2n-1$ тривимірні обмеження завантаження перевіряються завжди, щоб запобігти створенню неприпустимого маршруту. Тоді може виникнути ситуація, коли всі часткові маршрути t^{2n-2} не будуть задовольняти тривимірним обмеженням завантаження, і це виявиться лише на останньому

рівні $i = 2n\text{-}1$. Таким чином, застосування евристики *H3* може привести до ситуації, коли розв’язок не буде отримано.

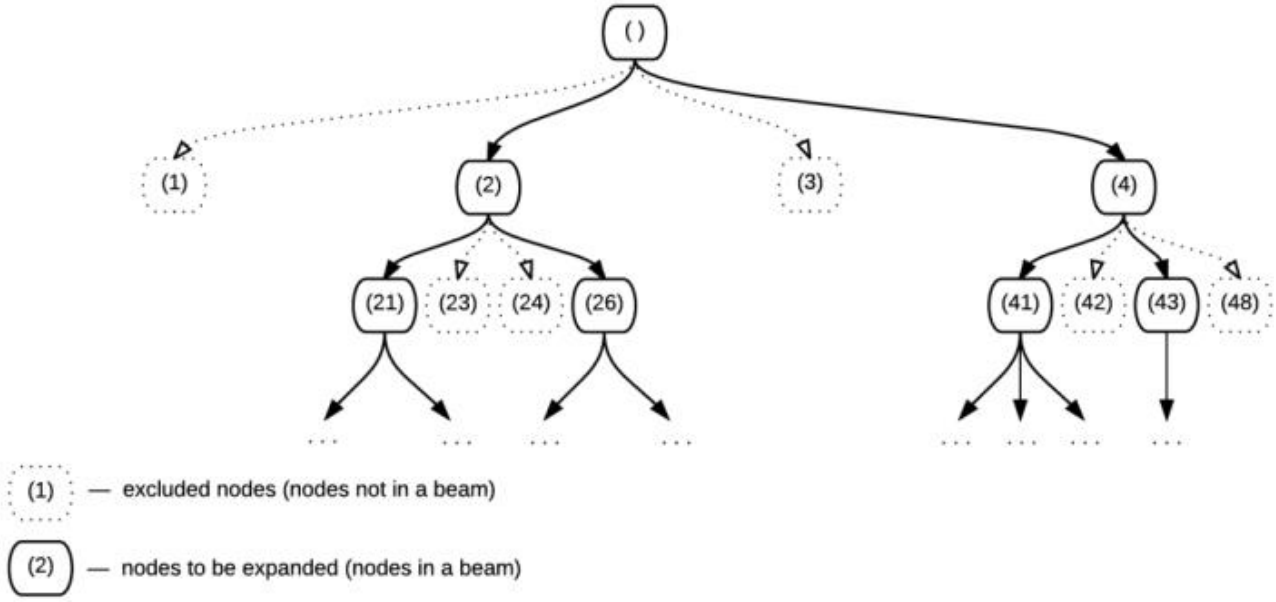


Рисунок 3.7. Фрагмент рекурсивного дерева, збудованого у процесі роботи алгоритму *GenBase* для $n = 4$

Наведемо приклад такої ситуації: величина *RBW* є настільки малою, що на кожному рівні обирається для подальшого розглядання тільки один пункт (наприклад, *RBW* = 1%), а ймовірність *check_prob* також дуже мала (наприклад, *check_prob*=0.01), тому тривимірні обмеження завантаження перевіряються порівняно рідко. Таким чином, алгоритм *GenBase* може генерувати часткові маршрути, що не задовольняють тривимірним обмеженням завантаження, і розпізнавати цей факт тільки при виконанні обов'язкової перевірки тривимірних обмежень на останньому рівні $i = 2n\text{-}1$. Тільки тоді стане зрозуміло, що генерований маршрут був неправильним.

Описаної ситуації можна уникнути, встановивши відповідні значення *check_prob* і *RBW*. На практиці, зазвичай, достатньо встановити *check_prob* = 0.2.

3.2 Задача складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції з призначенням поїздів на залізничні колії (TYSP)

У даному пункті розглянуто задачу оптимізації призначення поїздів на залізничні колії при обробці вантажів. Побудовано математичну модель і метод розв'язання досліджуваної задачі. Розроблена математична модель описує задачу в термінах комбінаторних конфігурацій на відміну від інших робіт [221]–[223], що використовують булеві змінні, та враховує призначення поїздів на залізничні колії, що у сукупності дозволяє знизити розмірність задачі та підвищити ефективність її розв`язання. Множина допустимих розв'язків розглядуваної задачі описується за допомогою композиційного *k*-образу комбінаторних множин – перестановок розміщень. Описаний в даному пункті метод розв'язання задачі TYSP ґрунтується також на комбінаторній генерації.

3.2.1 Постановка задачі

Опишемо задачу TYSP в оригінальній постановці, наведеній в [221]. Задано множину поїздів $I = \{1, 2, ..., N\}$ (кожному поїзду відповідає номер від 1 до N), кожен з яких везе контейнери, які потрібно перевантажити в один чи декілька інших (цільових) поїздів і/або має отримати контейнери від одного чи декількох поїздів. Матриця $A_{N\times N}$ задає кількість контейнерів, що поїзд i везе до поїзда j, $i, j = 1, 2, ..., N$. При цьому переміщення контейнерів з одного поїзда в інший здійснюється на сортувальній станції за допомогою портальних кранів. Сортувальна станція має G колій (G значно менше N) і працює ітераційно, в один проміжок часу, що зветься сервісним слотом (service slot), приймаючи максимум G поїздів. Необхідно знайти оптимальний за вартістю розклад прибуття поїздів та обробки вантажів на сортувальній станції.

При цьому можливі такі ситуації:

- повторні відвідування (revisit), коли вже обслужений поїзд повинен повторно заходити на сортувальну станцію (щоб прийняти контейнери, що призначалися йому, але прибули після першого обслуговування поїзда)
- розділені переміщення вантажу (split moves), коли поїзд *i*, що перевозить контейнери для поїзда *j*, обслуговується в сервісному слоті *t*, що передує сервісному слоту *t'* поїзда *j*. В останньому випадку контейнери з поїзда *i* тимчасово переміщуються на складський майданчик (storage area), звідки потім перевантажуються до поїзда *j* в сервісному слоті *t'*.

В роботі [221] описується п’ять рівнів деталізації розв’язання задачі TYSP:

1. Розбиття поїздів на групи і складання графіка їх прибуття на сортувальну станцію (формування сервісних слотів).

2. Призначення поїздів на залізничні колії.

3. Прийняття рішення про розташування контейнерів в поїздах.

4. Прив'язка переміщення контейнерів до портальних кранів (gantry cranes).

5. Прийняття рішення про послідовність переміщень контейнерів портальними кранами.

Робота [221] описує лише розв’язання першого рівня проблеми – формування сервісних слотів. На цьому рівні розв’язанням задачі є послідовність усіх сервісних слотів. При цьому для кожного поїзда *i* враховуються обмеження на найраніший e_i та найпізніший l_i допустимі слоти прибуття. Схематичне зображення сортувальної станції з [221] наведено на рис. 3.8.

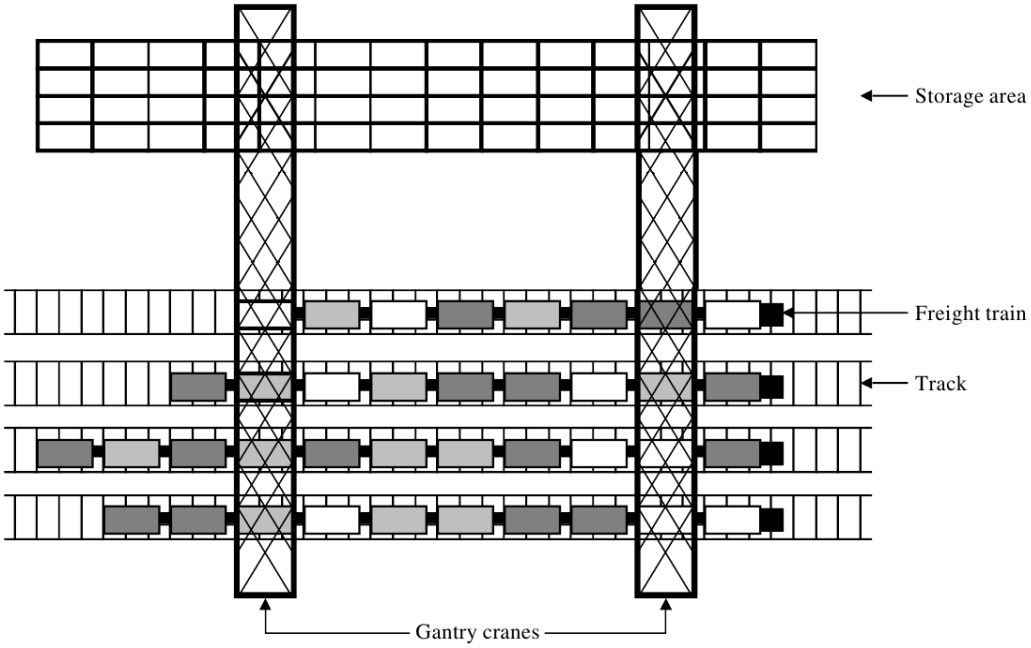


Рисунок 3.8. Схематичне зображення сортувальної станції [221]

3.2.2 Математична модель задачі TYSP2

Розглянемо другий рівень деталізації задачі TYSP [221]. Згідно другого рівня деталізації, при формуванні сервісних слотів поїздів приймається рішення, на яких коліях їх розмістити:

- якщо вихідний і цільовий поїзди обслуговуються в одному і тому самому сервісному слоті, доцільно розмістити їх на якомога ближчих одна до одної залізничних коліях;
- якщо поїзди обслуговуються в різних сервісних слотах, важливо розмістити вихідний і цільовий поїзди таким чином, щоб рухи портального крана були якомога коротшими. В цьому випадку кран повинен спочатку перемістити контейнери з вихідного поїзда на складський майданчик, а потім зі складського майданчика в цільовий поїзд, тому як вихідний, так і цільовий поїзди доцільно розмістити на коліях, якомога ближчих одна до одної.

Таким чином, врахування призначення поїздів на залізничні колії дозволяє знизити затрати на функціонування сортувальної станції, а саме на роботу портальних кранів при перевантаженні контейнерів.

Позначимо задачу, що враховує другий рівень деталізації TYSP, як TYSP2 (Transshipment Yards Scheduling Problem, level 2). Задача TYSP2 полягає в знаходженні оптимального (згідно функції цілі) розкладу прибуття вантажних поїздів та обробки вантажів на сортувальній станції з урахуванням призначення поїздів на залізничні колії при заданих:

- кількості колій на сортувальній станції;
- кількості контейнерів, що везе кожний поїзд кожному;
- обмежень на найраніший та найпізніший час (сервісний слот) прибуття кожного поїзда;
- вартості переміщення контейнерів портальним краном.

Побудуємо математичну модель задачі TYSP2. Опишемо кожен сервісний слот $t = 1,2,...,T$ за допомогою кортежу $K^t = (k_1^t, k_2^t, ..., k_g^t, ..., k_G^t)$, що містить номери всіх поїздів, що обслуговуються в цьому слоті; при цьому порядок поїздів є важливим і визначає їх призначення на колії. Тут $k_g^t \in I,\ g = 1,2,...,G$ – поїзд, що обслуговується в слоті *t* і розміщений на колії *g*.

Послідовність всіх сервісних слотів $K^1, K^2, ..., K^T$ описує розклад прибуття поїздів та обробки вантажів на сортувальній станції. Кожен сервісний слот формується в результаті вибору поїздів та призначення їх на колії (тобто впорядкування), тому кожен сервісний слот K^t обирається з множини розміщень $A_N^G : K^t \in A_N^G, t = 1,2,...,T$. Розв'язання задачі TYSP2 визначається побудованим кортежем $K = (K^1, K^2, ..., K^T)$.

В пункті 1.3 було відмічено, що множину допустимих розв'язків задачі [221] можна описати за допомогою кортежу сполучень. Аналогічно, множину допустимих розв'язків задачі TYSP2 можна описати за допомогою іншої *k*-множини – кортежу розміщень.

За аналогією з (3.1)-(3.2), здійснимо занурення f' кортежів K^t, $t = 1,2,...,T$ в евклідів простір.

$$
\begin{aligned}
&f': A_N^G \to R^G, \forall K^t = (k_1^t, k_2^t ... k_G^t) \in A_N^G, \\
&\mathrm{K}^t = f'(K^t) = (\kappa_1^t, \kappa_2^t ... \kappa_G^t) \in E'^G_N \subset R^G, \\
&\kappa_g^t = k_g^t, g = 1,2,...,G, \\
&t = 1,2,...,T.
\end{aligned}
\tag{3.14}
$$

У результаті відображення f' кожному елементу множини A_N^G буде поставлена у відповідність точка множини E'^G_N, що належить евклідовому простору R^G. Розв'язком задачі TYSP2 є вектор, що містить усі сервісні слоти

$$
X = (\mathrm{K}^1, \mathrm{K}^2, ..., \mathrm{K}^T) \in \mathbb{R}^{G \cdot T}. \tag{3.15}
$$

В загальному випадку кожен розклад можна оцінювати за багатьма критеріями, тому будемо вважати задачу TYSP2 багатокритеріальною задачею комбінаторної оптимізації, де критеріями є:

CR1. Мінімальна кількість повторних відвідувань поїздів (revisits).

CR2. Мінімальна вартість розділених переміщень контейнерів (split moves), яка залежить в тому числі від призначення поїздів на залізничні колії.

CR3. Мінімальна вартість «прямих» переміщень контейнерів між поїздами, що обслуговуються в одному і тому ж сервісному слоті, що також залежить в тому числі від призначення поїздів на колії.

Критерій CR1 повторює аналогічний критерій в [221]; критерій CR2 є розширенням критерію [221]; критерій CR3 є новим порівняно з [221], оскільки, на відміну від [221], дана постановка задачі враховує призначення поїздів на залізничні колії.

Опишемо основні обмеження:

$$
e_{\kappa_g^t} \le t \le l_{\kappa_g^t}, t = 1,2,...,T,\ g = 1,2,...,G \tag{3.16}
$$

$$
\kappa_p^t, \kappa_q^{t'} : t \le t' + y_{\kappa_q^{t'}} \cdot M,\ \ t = 1,2...T, t' = t, t+1, ..., T,\ p,q = 1,2,...,G, \tag{3.17}
$$

де T – кількість сервісних слотів (індекс t);

G – кількість колій на сортувальній станції;

e_i та l_i – відповідно найраніший та найпізніший сервісні слоти, в які припустимо обслуговувати поїзд i;

M – деяке велике ціле число (наприклад, $M = T - 1$);

y_i – булева змінна, що залежить від X та дорівнює 1, якщо поїзд i повторно відвідує сортувальну станцію (revisit), тобто входить до 2 сервісних слотів $t \neq t'$:

$$y_i = \begin{cases} 1, if\ \exists t, t', g, g' : t \neq t', \kappa_g^t = \kappa_{g'}^{t'} = i \\ 0\ otherwise \end{cases}$$

Умова (3.16) гарантує, що кожний поїзд обслуговується не раніше слоту e_i та не пізніше слоту l_i.

Аналогічно умові (4) в [221], умова (3.17) забезпечує прибуття вихідного поїзда κ_p^t перед цільовим поїздом $\kappa_q^{t'}$; проте, якщо цільовий поїзд буде заїжджати повторно (revisit), то умова завжди задовольняється завдяки доданку $y_{\kappa_q^{t'}} \cdot M$.

Введемо також наступні позначення:

$z_{tt'} = \begin{cases} 1, if\ t < t' \\ 0, if\ t = t' \end{cases}$ – булева змінна, що дорівнює 1, якщо сервісні слоти t та t' є різними, та 0 – якщо ідентичними;

C_{pq} – вартість переміщення портальним краном одного контейнера з поїзда, що розміщений на колії p на поїзд, що розміщений на колії q, якщо поїзди обслуговуються в одному і тому ж часовому інтервалі;

C_{pq}^* – вартість переміщення портальним краном одного контейнера з поїзда, що розміщений на колії p на поїзд, що розміщений на колії q, якщо

поїзди обслуговуються в різних часових інтервалах (вартість роздільного руху);

C^r – вартість одного повторного відвідування (revisit).

Величини C_{pq} та C^*_{pq} є незмінними і можуть бути обчислені один раз, на початку роботи методу розв'язання задачі:

$$C_{pq} = c_{pq} \tag{3.18}$$

$$C^*_{pq} = c_{p0} + c_{0q}, \tag{3.19}$$

де c_{pq} – фіксована вартість переміщення контейнера з колії p на колію q.

Складський майданчик позначається фіктивною колією 0. Тому, якщо контейнер переміщується безпосередньо з поїзда в поїзд, вартість C_{pq} дорівнює вартості прямого переміщення контейнера c_{pq}. Однак, якщо поїзди обслуговуються в різних слотах, то контейнер спочатку доведеться перемістити з вихідного поїзда в складський майданчик (c_{p0}), а потім, пізніше, зі складського майданчику в цільовий поїзд (c_{0q}), тому

$$C^*_{pq} = c_{p0} + c_{0q}.$$

Всі константи c_{pq} залежать від особливостей конкретної сортувальної станції. У найпростішому випадку, якщо вартість переміщення одного контейнера залежить тільки від відстані між залізничними коліями, а сусідні колії знаходяться на однаковій відстані, можна прийняти

$$c_{pq} = |p - q|.$$

Побудуємо узагальнений критерій ефективності задачі TYSP2, що є згорткою критеріїв CR1-CR3. У формулі (3.20) перший доданок відповідає критерію CR1, другий – критерію CR2, третій – критерію CR3.

$$\Psi'(X) = \alpha_1 \sum_{i \in I} C^r y_i + \alpha_2 \sum_{t=1}^{T} \sum_{t'=t+1}^{T} \sum_{p=1}^{G} \sum_{q=1}^{G} z_{tt'} A_{\kappa_p^t \kappa_q^{t'}} C^*_{pq} + \alpha_3 \sum_{t=1}^{T} \sum_{p=1}^{G} \sum_{q=1}^{G} A_{\kappa_p^t \kappa_q^t} C_{pq}, \quad (3.20)$$

де $I = \{1,2, ..., N\}$ – множина поїздів

$A = [A_{ij}]$, $i,j = 1,2, ..., N$ – кількість контейнерів, які поїзд i перевозить для поїзда j;

$\alpha_1, \alpha_2, \alpha_3$ – вагові коефіцієнти критеріїв оптимізації CR1-CR3.

Математична модель задачі TYSP2 може бути представлена як

$$\begin{gathered} \min_{X \in \mathbb{W}} \Psi'(X) \\ \mathbb{W} = \{X \in \mathbb{R}^{G \cdot T}\} \\ X = (\mathrm{K}^1, \mathrm{K}^2, ..., \mathrm{K}^T) \in \mathbb{R}^{G \cdot T}, \\ \mathrm{K}^t = (\kappa_1^t, \kappa_2^t ... \kappa_G^t) \in E'^G_N \subset R^G, \kappa_g^t \in I, g = 1,2,...,G \\ t = 1,2...T, \end{gathered} \quad (3.21)$$

де допустимі розв’язання X є зануренням кортежу розміщень у евклідів простір, задовольняють обмеження (3.16)-(3.17).

3.2.3 Метод розв’язання задачі TYSP2

Як і в роботі [221], запропонований метод розв’язання задачі TYSP2 використовує евристику променевого пошуку.

Опишемо стратегію розв’язання задачі TYSP2, яку реалізує алгоритм, що є розвитком алгоритму розв’язання задачі TYSP, описаного в [221].

1. На першому етапі приймемо $t = 0$. Це відповідає порожньому вузлу на графі [221]. Назвемо його батьківським вузлом.

2. Величина t збільшується на одиницю, і генерується множина всіх можливих сервісних слотів $\mathrm{K}^t = (\kappa_1^t, \kappa_2^t ... \kappa_G^t) \in E'^G_N$.

3. Обчислюється значення цільової функції (3.20) для кожного згенерованого слоту і серед них обирається *BW* (ширина променя, наперед заданий параметр) кращих з точки зору (3.20).

4. Для кожного з обраних слотів створюється новий вузол в ациклічному графі (або рекурсивному дереві) [221], який знаходиться під батьківським вузлом, і здійснюється рекурсивний перехід до кроку 2, встановивши щойно збудований вузел в якості батьківського.

Алгоритм [221] працює так само, як і алгоритм *GenBase*, будуючи у процесі своєї роботи рекурсивне дерево (ациклічний граф в термінології [221]). Тому розв'язання описаної задачі може бути здійснене за допомогою алгоритму *GenBase*.

Опишемо роботу алгоритму *GenBase*, застосовуючи термінологію задачі TYSP2. Повним розв'язанням задачі є набір з T сервісних слотів. На кожному рівні $t = 1,2,...,T$ алгоритм *GenBase* формує сервісний слот K^{t+1}, додаючи його до поточного часткового розв'язання, що містить *t* сервісних слотів $\mathrm{K}^1,\mathrm{K}^2,...,\mathrm{K}^t$. На рівні $i=T$ алгоритм отримує повний розв'язок задачі, тобто набір з *T* сервісних слотів.

Процедура *GetF_TSY,* що використовується для формування кортежу F^i та передається в алгоритм *GenBase*, наведена нижче. Цільова функція (3.20) передається з множиною параметрів *p* як *target_func*. До множини параметрів входять також кількість колій *G*, множина поїздів *I*, ширина променя *BW* (beam width, параметр евристики променевого пошуку), найраніші $\{e_1,e_2,...,e_{|I|}\}$ та найпізніші $\{l_1,l_2,...,l_{|I|}\}$ допустимі сервісні слоти для всіх похздів $i \in I$.

Зазначимо, що процедура *GetF_TSY* використовує процедуру для отримання всіх розміщень, що позначається як *make_arrangements*. Такі розміщення можуть бути отримані за допомогою відомих методів, наприклад [25]. У мові Python такі розміщення можуть бути отримані за допомогою стандартного модуля itertools [229].

```
def GetF_TSY ((K^1,K^2,...,K^t),
                p={G, I, target_func, BW, {e_1,e_2,...,e_|I|}, {l_1,l_2,...,l_|I|} })
  available_trains = [ ]
  for i in | I |
    if e_i < t < l_i
      available_trains.append(i)
  all_slots = make_arrangements(available_trains, G)
  sorted_slots = sorted(all_slots, key= target_func)
  best_slots = sorted_slots[BW:]
  return best_slots
```

Приклад. На рис. 3.9 наведено рекурсивне дерево, побудоване у процесі роботи алгоритму *GenBase* під час розв'язання задачі TYSP2 для $G = 2$ колій, 4 поїздів $I = \{1,2,3,4\}$ і ширині променя $BW = 2$. Рекурсивне дерево відповідає ациклічному графу з [221]; відміна лише в тому, що текст вузла в даному дереві вказує на поточний слот замість множини поїздів, вже включених у розклад на поточний момент, як це було в [221]. Вузли, обрані евристикою променевого пошуку для подальшого розширення (expand), виділені жирним шрифтом, а пропущені вузли замінені на «...».

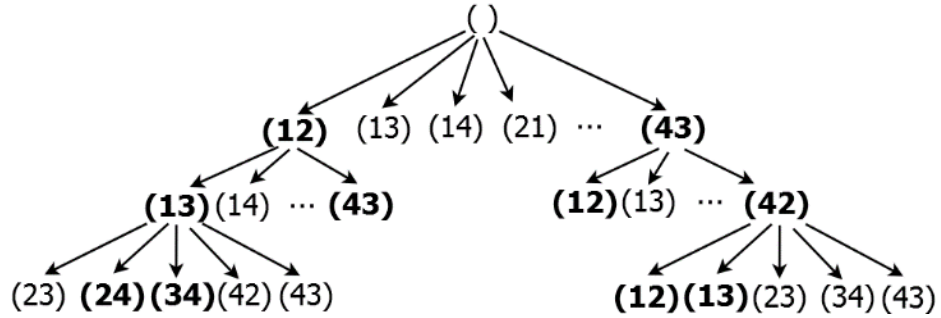


Рисунок 3.9. Приклад ациклічного графа розв'язання

На нульовому рівні $t = 0$ є лише один пустий вузел ( ). Наступний рівень $t = 1$ містить всі можливі варіанти слотів $K^1 = (\kappa_1^1, \kappa_2^1) \in E'^2_4$, тобто {(12), (13), ... (43)}. Слід зазначити, що слоти (12) і (21) розглядаються як різні, оскільки для даної постановки задачі TYSP2 призначення поїздів на колії має значення. Наприклад, якщо контейнери з поїзда 2 мають бути перевантажені на складський майданчик (що позначається фіктивною колією 0), то доцільніше

розмістити поїзд 2 на першій колії (що ближче до складського майданчику), ніж на другій. Це означає, що слот (21) буде мати меншу *вартість початку розділеного переміщення*, ніж (12).

Після генерації $\left|A_4^2\right| = 12$ всіх можливих слотів K^1 за допомогою евристики променевого пошуку обирається $BW = 2$ вузлів (сервісних слотів) з кращим значенням цільової функції для подальшого розширення. Інші вузли будуть виключені з подальшого розгляду. Припустимо, були вибрані слоти (12) та (43).

Наступний рівень $t = 2$ містить можливі варіанти слотів для батьківських слотів $K^1 = (12)$ та $K^1 = (43)$. Зазначимо, що слоти K^1 і K^2 можуть містити один і той самий поїзд (наприклад, $K^1 = (12)$ і $K^2 = (13)$ містять один і той самий поїзд 1). Нехай сервісні слоти $K^2 = (13)$, $K^2 = (43)$ для батьківського слоту $K^1 = (12)$ та $K^2 = (12)$, $K^2 = (42)$ для батьківського слоту $K^1 = (43)$ є найкращими з точки зору (3.20) Це означає, що на даний момент є чотири розв’язання (розклади руху): (12) (13), (12) (43), (43) (12) та (43) (42). Розклади (12) (43) та (43) (12) вже містять всі поїзди, тому вони позначаються як завершені.

Далі необхідно сформувати третій сервісний слот для розкладів (12) (13) та (43) (42). Нехай кращими сервісними слотами для $t = 3$ є (24) та (34) для лівої частини графа, а також (12) та (13) для правої частини графа на рис. 3.9. Отже, отримано чотири нові повні розклади: (12) (13) (24), (12) (13) (34), (43) (42) (12) та (43) (42) (13). Таким чином, алгоритм розв’язання згенерував вісім повних розв’язків (розкладів руху), і розклад з кращим значенням цільової функції буде кращим розв’язком.

3.3 Висновки по третьому розділу

В розділі описано застосування методів комбінаторної генерації в розв'язанні деяких класів задач перевезення та обробки вантажів. Для задачі

вивозу і доставки (Pickup and Delivery Problem, PDP) розроблено математичну модель, що використовує комбінаторні конфігурації замість булевих змінних, які застосовуються в інших роботах. Стратегії та методи розв'язання задачі PDP набули подальшого розвитку завдяки використанню узагальненого методу комбінаторної генерації та врахуванню додаткових властивостей задачі (тривимірних обмежень завантаження, запобігання блокування та перевірки стійкості), що у сукупності дозволяє підвищити ефективність розв`язання задачі. Основними перевагами запропонованої стратегії розв'язання є можливість балансування між точністю отриманого евристичного розв'язання і часом роботи алгоритму шляхом зміни відносної ширини променю *RBW*, можливість отримання декількох альтернативних варіантів розв'язання (оскільки в алгоритмі *GenBase* на останньому рівні може бути згенеровано декілька повних маршрутів з одного вхідного часткового маршруту) та гнучкість методу (шляхом підстановки іншої процедури *GetF* можна легко змінити логіку отримання пунктів-кандидатів). Побудовано математичну модель та метод розв'язання задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції (TYSP2). На відміну від інших робіт, що використовують булеві змінні для опису задачі TYSP2, математична модель використовує математичний апарат комбінаторних множин. Множина допустимих розв'язків досліджуваної задачі описується за допомогою *k*-множини – кортежу розміщень. Описана постановка задачі TYSP2 розширює оригінальну задачу [221], розглядаючи додатково другий рівень деталізації, тобто ураховуючи призначення поїздів на колії, що дозволяє знизити вартість роботи портальних кранів на сортувальних станціях. Розвинуто метод розв'язання досліджуваної задачі завдяки використанню узагальненого методу генерації комбінаторних конфігурацій.

Результати розділу опубліковано в роботах [40], [41], [53]–[56].

4 ОПИС ПРОГРАМ ТА ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ

У даному розділі описується розробка програмного комплексу для розв’язання задач комбінаторної генерації та комбінаторної оптимізації, описаних у попередніх розділах, а також обчислювальні експерименти та порівняння отриманих результатів з результатами існуючих досліджень.

Розроблене програмне забезпечення містить наступні модулі:

- модуль генерації композиційних *k*-образів комбінаторних множин (див. пункт 2.2);
- модуль генерації перестановок з частково заданою сигнатурою (див. пункт 2.4);
- модуль розв’язання задачі Pickup and Delivery Problem з тривимірними обмеженнями завантаження, описаної в пункті 3.1;
- модуль розв’язання задачі складання розкладу руху вантажних поїздів з розмішенням поїздів на коліях (див. пункт 3.2);

Структуру розробленого програмного комплексу наведено на рис. 4.1.

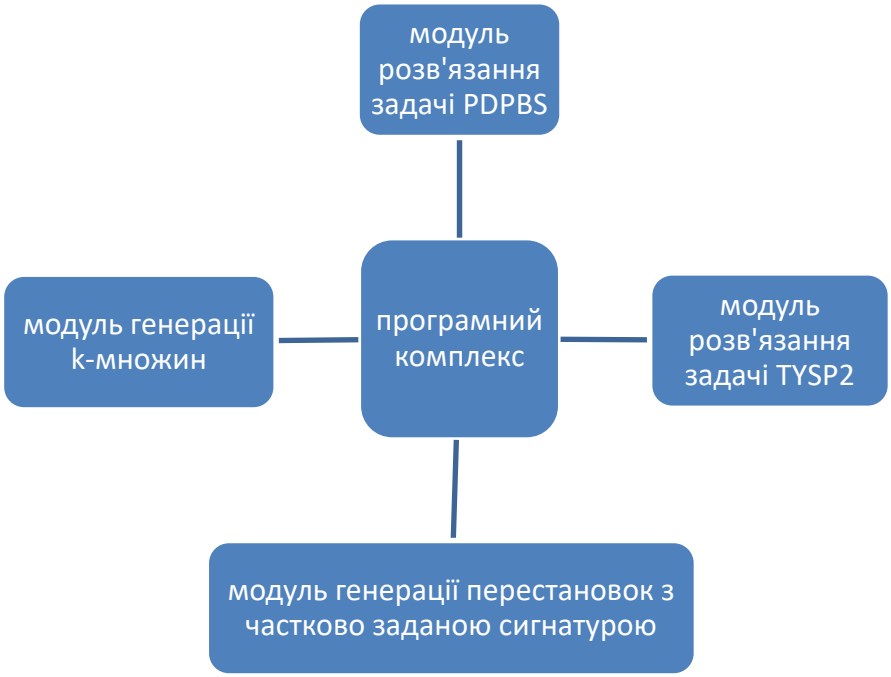


Рисунок 4.1. Структура розробленого програмного комплексу

У наступних пунктах наведено детальний опис кожного модуля розробленого програмного забезпечення.

4.1 Модуль генерації *k*-множин

На основі запропонованого алгоритму розв'язання задачі генерації композиційних $k-$образів комбінаторних множин (див. пункт 2.2.3) було розроблено програмний модуль, за допомогою якого можна генерувати $k-$ множини різноманітної структури і складності. Програмний модуль був розроблений на мові Delphi 7.

Опишемо результат роботи розробленого програмного модуля для $k-$ множини з прикладу 2.2.

В кінці нульової ітерації алгоритму *Gen_k-set* отримаємо

$$P^1 = \{(cdfg),(cdgf),(cefg),(cegf),(defg),(\deg f),$$
$$(f\ \gcd),(gfcd),(fgce),(gfce),(fgde),(gfde)\}.$$

Тут останні 6 кортежів являють собою перестановки пар в перших шістьох кортежах, що відповідають перестановкам елементів *a* та *b* у множині Y_0.

В кінці першої ітерації *Gen_k-set* отримаємо

$$P^2 = W_z = \{(123456xywz),(123456wzxy),(123789xywz),(123789wzxy),$$
$$(456789xywz),(456789wzxy),(xywz123456),(wzxy123456),$$
$$(xywz123789),(wzxy123789),(xywz456789),(wzxy456789)\}.$$

За формулою (2.9) отримаємо, що для генерації базових множин знадобилося $(2\cdot 2)+(2\cdot 3+2\cdot 2)+(3\cdot 1+3\cdot 1+3\cdot 1+3\cdot 1+3\cdot 1)=29$ викликів *GenBase*. За формулою (2.11) можна підрахувати, шо виклик *replace* проводився $(2\cdot(3\cdot 2))+(12\cdot(1\cdot 1\cdot 1\cdot 1\cdot 1))=24$ разів. Отримана $k-$множина містить 12 елементів, що співпадає з підрахунками за формулою (2.5).

Скріншоти програмного модуля наведено на рис. 4.2 та 4.3.

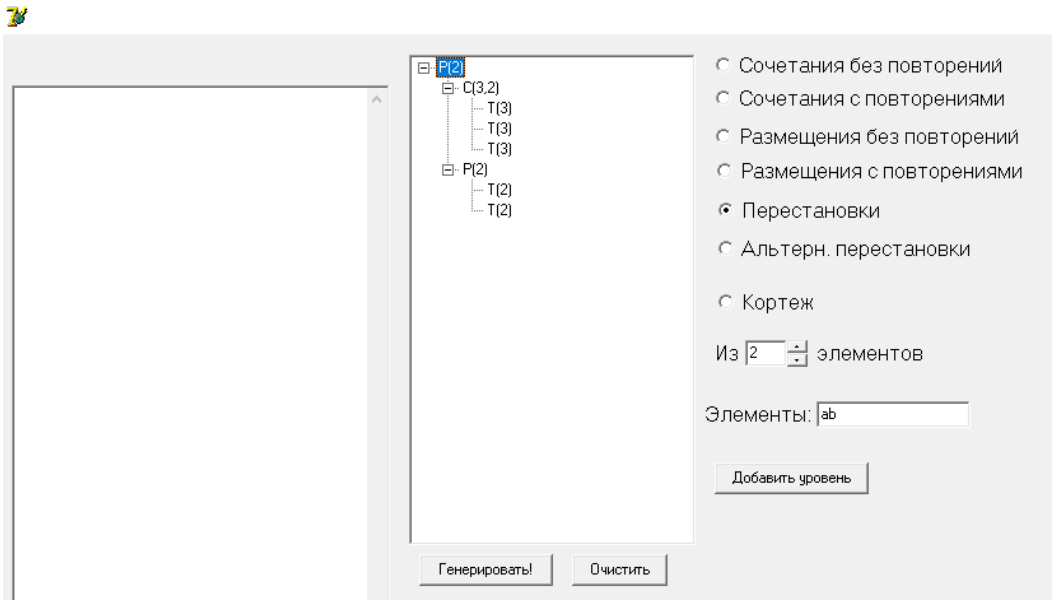


Рисунок 4.2. Вхідні дані для прикладу 2.2

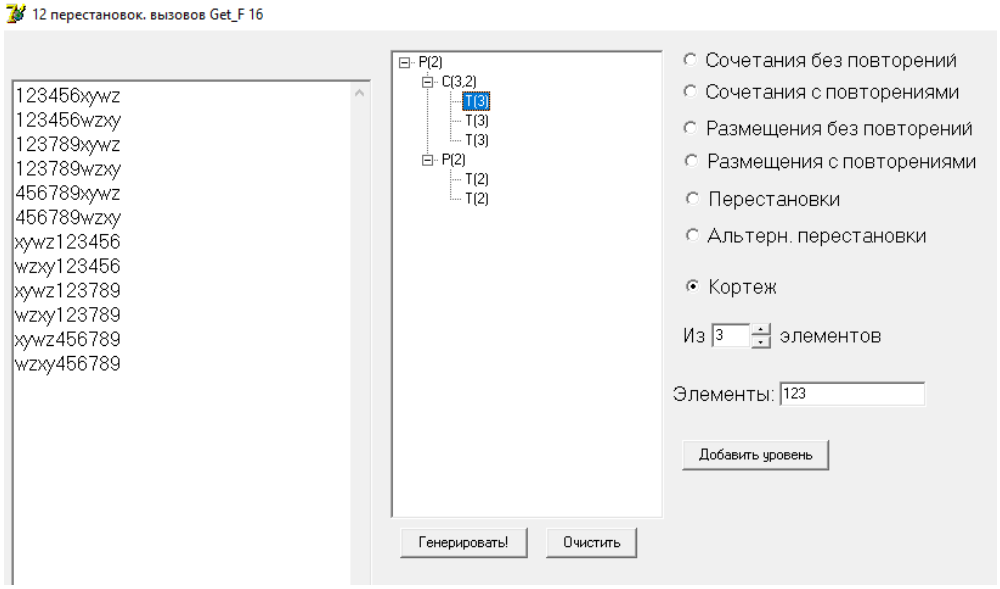


Рисунок 4.3. Згенерована *k*-множина для прикладу 2.2

4.2 Модуль генерації перестановок з частково заданою сигнатурою

Алгоритм генерації перестановок з частково заданою сигнатурою (див. пункт 2.4) було реалізовано на мові Delphi 7 як модифікацію модуля генерації *k*-множин (див. пункт 4.1). Отриманий програмний модуль дозволяє генерувати перестановки з будь-якими частково заданими сигнатурами $r(\pi)$.

Скріншот вхідних даних для прикладу 2.6, введених у розроблений програмний модуль, наведено на рис. 4.4. Скріншот результатів генерації перестановок з частково заданою сигнатурою з прикладу 2.6 наведено на рис. 4.5.

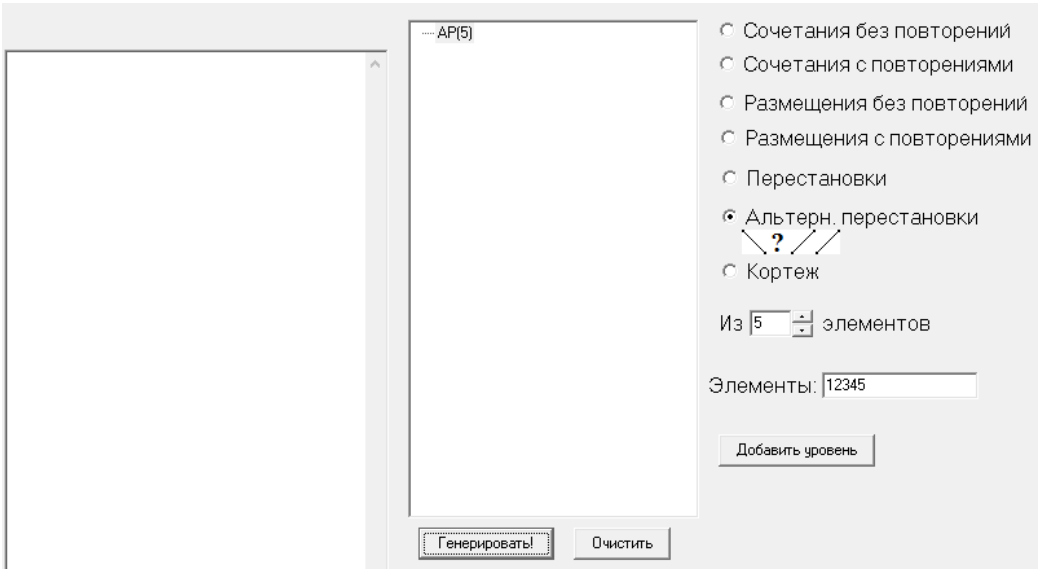


Рисунок 4.4. Вхідні дані для прикладу 2.6

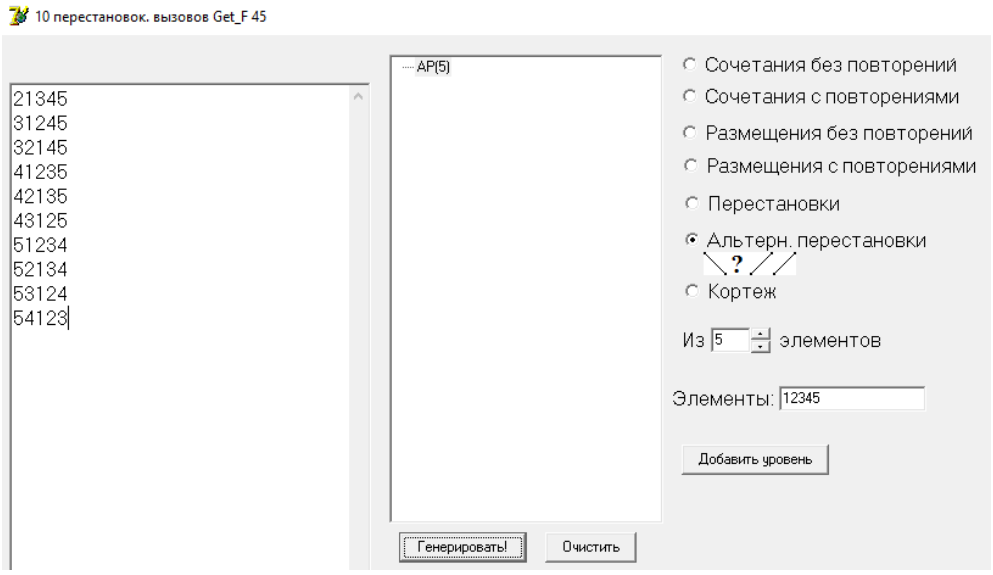


Рисунок 4.5. Згенеровані перестановки з частково заданою сигнатурою з прикладу 2.6

4.3 Модуль розв’язання задачі PDPBS

У даному пункті наведено опис програмного модуля розв’язання задачі PDPBS, описаної в пункті 3.1, та результати обчислювальних експериментів, що містять порівняння отриманих результатів з результатами, отриманими в інших дослідженнях.

4.3.1 Опис програмного модуля

Розроблено програмний модуль розв’язання задачі PDPBS, що має зручний інтерфейс. Програмний модуль доступний онлайн за адресою https://pdp-app.000webhostapp.com/html/ .

Демонстрацію роботи програмного модуля також можна побачити на https://rebrand.ly/pdp-demo-video.

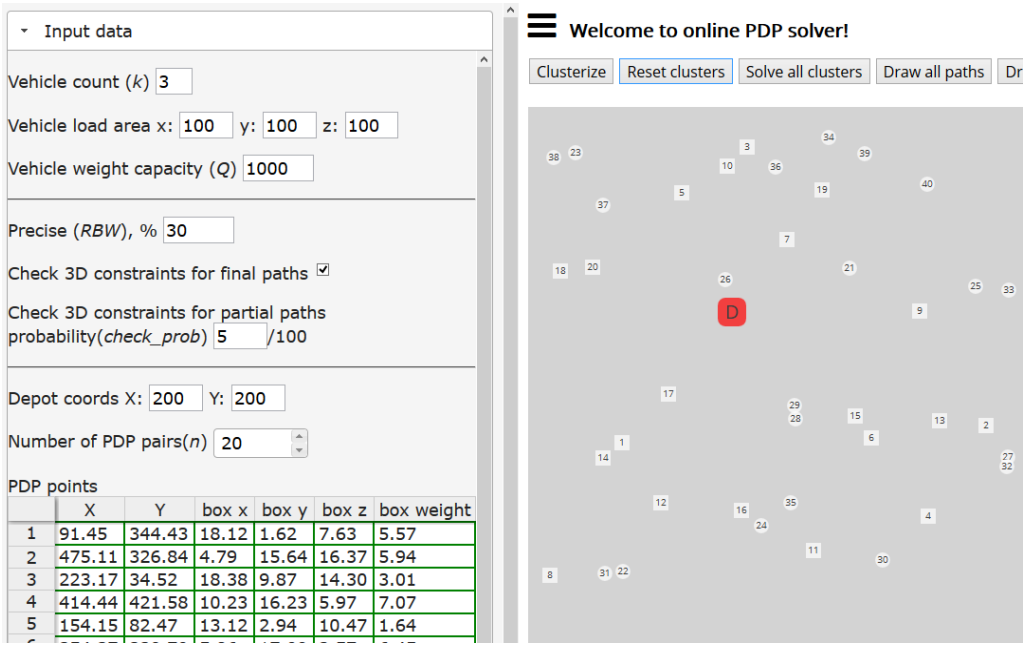
Вхідними параметрами програмного модуля є:

1) параметри транспортного засобу:
   - кількість транспортних засобів *v*;
   - лінійні розміри *L, W, H* вантажного відсіку і вантажопідйомність *Q* кожного транспортного засобу;

2) параметри пунктів:
   - координати депо;
   - кількість пар pickup-delivery;
   - координати кожного пункту;
   - маса та розмір контейнерів, що перевозяться;

3) параметри розв’язання:
   - *RBW,%* – відносна ширина променю;
   - *check_prob* – ймовірність перевірки тривимірних обмежень завантаження для неповного маршруту.

На рис. 4.6 наведено скріншот програми з прикладом вхідних даних.



	X	Y	box x	box y	box z	box weight
1	91.45	344.43	18.12	1.62	7.63	5.57
2	475.11	326.84	4.79	15.64	16.37	5.94
3	223.17	34.52	18.38	9.87	14.30	3.01
4	414.44	421.58	10.23	16.23	5.97	7.07
5	154.15	82.47	13.12	2.94	10.47	1.64

Рисунок 4.6. Скріншот прикладу вхідних даних задачі (маленькі квадрати позначають пункти pickup, кола – пункти delivery, великий квадрат – депо)

Програмний модуль дозволяє:

1) сформувати кластери, тобто розв’язати задачу на першому етапі, розподіливши усі пари pickup-delivery між транспортними засобами;

2) отримати евристичний розв’язок для кожного транспортного засоба (кластера), тобто розв’язати задачу на другому етапі.

Приклад результату кластеризації 20 пар pickup-delivery наведено на рис. 4.7. Квадратами позначені пункти pickup, колами – пункти delivery, а депо позначене великим закругленим квадратом.

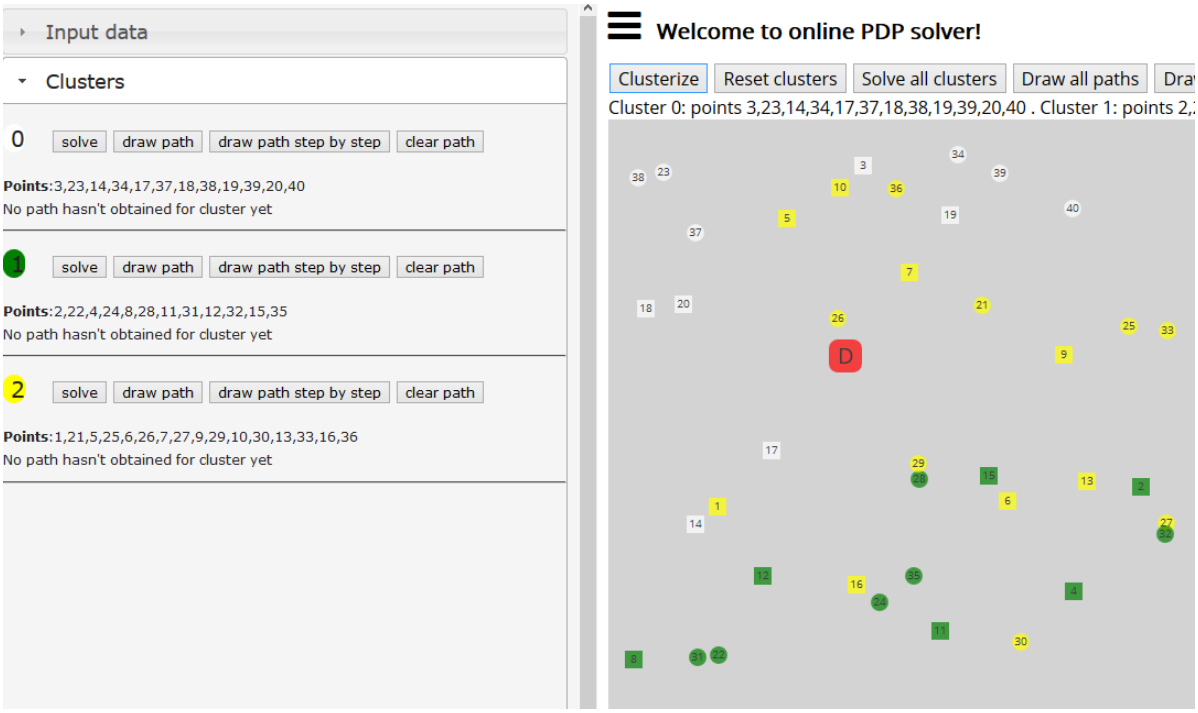


Рисунок 4.7. Скріншот прикладу кластеризації задачі на 20 пар pickup-delivery

На рис. 4.8 представлено скріншот маршрутів, згенерованих для кожного кластера з наведеного вище прикладу.

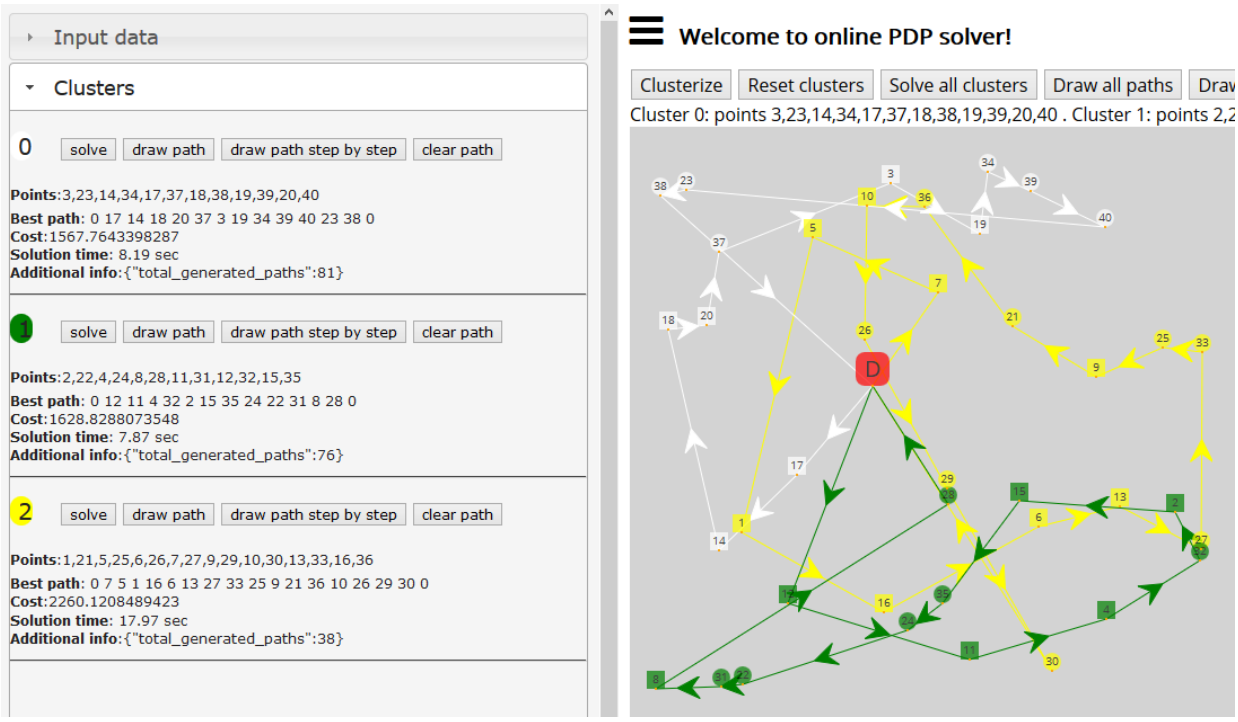


Рисунок 4.8. Скріншот результату побудови оптимальних маршрутів для всіх кластерів

4.3.2 Порівняння отриманих результатів з існуючими

У даному підпункті наводиться оцінка ефективності наведеного алгоритму розв'язання задачі PDPBS та порівняння результатів, отриманих в даній роботі, з результатами, отриманими в інших дослідженнях. Слід відзначити, що порівняння проводиться на другому етапі розв'язання задачі, тому що на першому етапі для формування кластерів використовується класичний алгоритм *k*-середніх з добре відомою оцінкою ефективності [230].

Обчислювальні експерименти проводилися на персональному комп'ютері Lenovo G575, AMD Dual-Core E-450 (1.65 ГГц), 4 GB RAM.

У таблицях 4.1-4.2 наведено результати обчислювальних експериментів. Було отримано кількість згенерованих маршрутів і вартість кращого маршруту при різних n та RBW (координати пунктів були згенеровані випадковим чином в діапазоні [0; 500]).

Результати наведено у форматі «загальна кількість згенерованих маршрутів» / «краща вартість маршруту». Наприклад, для $2n = 12$ і $RBW = 20\%$ було згенерувано 298 маршрутів, а вартість кращого маршруту склала 1869.

Отримані результати було порівняно з двофазною евристикою (two-phase heuristic), запропонованою Ренольдом [231] з параметрами $R = 5$ та різними значеннями α: $\alpha = 0{,}5; 1; 1{,}5$.

Варто відзначити, що постановка задачі [231] не враховує жодних обмежень на завантаження. Тому для коректного порівняння результатів даної роботи з результатами Ренольда при проведенні обчислювальних експериментів тривимірні обмеженя завантаження не перевірялися.

При порівнянні розв'язків, отриманих за допомогою описаної евристики, та розв'язків, отриманих за допомогою евристики [231], виявилося, що вартість маршрутів є співрозмірною з вартістю маршрутів, отриманих за допомогою двофазної евристики [231]. Проте, недоліком описаного алгоритму є значний час його роботи: в той час як алгоритм Ренольда працює всього кілька мілісекунд, алгоритм *GenBase* може працювати до 60 секунд для 25 пар pickup-delivery.

Таблиця 4.1

Порівняння результатів роботи алгоритму Ренольда з алгоритмом *GenBase*

2*n*	*RBW=1%*	*RBW=5%*	*RBW=10%*	*RBW=20%*	*20%<RBW<50%*	*Краша вартість маршруту*	*Середній час роботи алгоритму*	*Вартість маршруту, отриманого за допомогою алгоритму Ренольда*		
								α=0.5	*α =1*	*α=1.5*
8	1/2044	1/2044	1/2044	10/1621	180/1531	1531	<16ms	1656	1531	1531
10	1/1985	1/1985	1/1985	44/1869	1972/1664	1664	<500ms	1830	1750	1664
12	1/2090	1/2090	213/1869	298/1869	2125/1869	1869	<500ms	1946	1796	2099
14	1/2198	1/2198	1/2198	14/2148	2666/2148	2148	<1s	2125	2124	2267
16	1/2520	1/2520	130/2285	4474/2285	–	2285	<2s	2323	2228	2662

Таблиця 4.2

Порівняння результатів роботи алгоритму Ренольда з алгоритмом *GenBase* (великі розмірності)

2*n*	*RBW=1%*	*RBW=5%*	*RBW=10%*	*RBW=11%*	*Краща вартість маршруту*	*Середній час роботи алгоритму*	*Вартість маршруту, отриманого за допомогою алгоритму Ренольда с*		
							α=0.5	*α =1*	*α=1.5*
20	1/2404	1/2404	3155/2097	8921/2097	2097	<10s	2019	2019	2035
	ν=1%	*ν=3,7%*	*ν=4%*	*ν=4,2%*					
30	1/3517	41/2800	545/2650	3102/2624	2624	<20s	2588	2804	3295
	ν=1%	*ν=2,7%*	*ν=3%*	*ν=3,25%*					
40	1/3458	170/3402	3014/3402	9778/3222	3222	<40s	3526	3166	3129
	ν=1%	*ν=2%*	*ν=2,1%*	*ν=2,25%*					
50	1/3553	1/3553	586/3178	6429/3151	3151	<60s	3036	3398	3180

На жаль, окрім [231], автором не було знайдено статей, присвячених розв’язанню задачі Pickup and Delivery Problem з достатньо схожою постановкою та тестовими даними, що знаходяться у відкритому доступі. Тому було вирішено порівняти результати даної роботи з результатами Ропке [189], представленими в http://www.diku.dk/~sropke/.

Робота Ропке присвячена розв’язанню задачі PDP з багатьма транспортними засобами та часовими вікнами (Pickup and Delivery Problem

with Time Windows). У постановці задачі [189] також враховується вантажопідйомність кожного транспортного засобу Q. Тому при порівнянні результатів даної роботи з результатами [189] було використано одне і те саме значення Q в обох випадках.

Однак слід відзначити, що результати, отримані в даній роботі, не можуть бути чітко зіставлені з результатами Ропке, оскільки постановка задачі PDP, описана в даній роботі, і постановка задачі в роботі [189], відрізняються тим, що постановка задачі в даній роботі не враховує часові вікна (Time Windows).

Незважаючи на це, можна, принаймні, перевірити, чи мають співрозмірні результати, отримані в даній роботі, з результатами Ропке. В роботі [189] Ропке розглядає чотири типи наборів вхідних даних (instances): A, B, C та D, різниця між якими полягає у вантажопідйомності транспортних засобів та ширині часових вікон. Для порівняння результатів даної роботи з результатами Ропке було обрано групи C та D, де часові вікна є ширшими. У таблиці 4.3 наведено результати порівняння розв'язків, отриманих Ропке в [189], та розв'язків, отриманих в даній роботі за допомогою методу, описаного в пункті 3.1 для тестових прикладів з [189]. Під час розв'язання тестових задач було експериментально підібрано такі значення *RBW*, які дозволяли отримати порівняно добрі результати за досить короткий час (менше, ніж 25 секунд). Як було зазначено раніше, евристика променевого пошуку, застосована в даній роботі, дозволяє балансувати між точністю отриманого розв'язку та часом роботи алгоритму. Тому для кожного набору вхідних даних, встановлюючи більші або менші значення *RBW,* можна або отримати більш точний розв'язок за порівняно більший час, або менш точний розв'язок за порівняно менший час відповідно. Як і слід було очікувати, загальна вартість розв'язання, отриманого за допомогою методу, описаного в даній роботі, в більшості випадків менша, ніж у Ропке в [189] через те, що постановка задачі Pickup and Delivery Problem в даній роботі не враховує обмеження на часові вікна.

Порівняння результатів даної роботи та результатів Ропке *Таблиця 4.3*

Вхідні дані (instance) Ропке	Результати Ропке		Результати даної роботи			
	вартість	час роботи, с	вартість	час роботи, с	Кількість згенерованих маршрутів	*RBW*, %
DD30	1133	49	955	0,315	16	1
DD35	1210	99	1137	1,107	34	2
DD40	1352	136	1198	4,253	91	2
DD45	1483	132	1322	20,8	348	2
DD50	1600	105	1425	7,833	1165	1,7
DD55	1743	124	1518	21,684	189	1,5
DD60	1869	247	1716	5,144	32	1,2
DD65	2125	209	1939	4,837	23	1,1
DD70	2220	175	2184	1,786	7	1
DD75	2396	201	2291	2,232	7	1
CC30	1087	76	1035	5,058	297	3
CC35	1230	97	1172	12,823	468	2,5
CC40	1358	132	1205	8,22	191	2
CC45	1509	82	1404	2,065	34	1,7
CC50	1689	168	1613	2,669	35	1,8
CC55	1816	196	1730	12,134	108	1,3
CC60	2015	127	1823	12,111	86	1,3
CC65	2172	145	2024	14,776	80	1,1
CC70	2201	288	2159	11,675	49	1
CC75	2375	325	2327	16,105	54	0,8

Крім [189], в статті [202] також описується розв’язання задачі Pickup and Delivery Problem з тривимірними обмеженнями завантаження, але постановка задачі PDP, описана в [202], містить занадто багато додаткових обмежень і особливостей, які не розглядаються в даній роботі. Зокрема робота [202] розглядає:

- можливість перезавантаження контейнера, тобто розвантаження контейнера і завантаження його назад в інше положення всередині вантажного відсіку транспортного засобу в будь-якому пункті pickup або delivery;
- можливість завантаження декількох контейнерів у пунктах pickup;
- крихкість контейнера (box fragility), яка призводить до додаткового обмеження, що полягає в тому, що не можна ставити ніякі контейнери поверх крихких контейнерів.

Тому автор роботи не бачить можливості порівнювати результати,

отримані за допомогою методу розв’язання задачі PDP, запропонованого в даній роботі, з результатами, отриманими в роботі [202], оскільки постановки задачі є занадто різними, не дивлячись на зовнішню схожість.

4.3.3 Порівняння точних та евристичних розв’язків

Як було показано в попередньому пункті, не виявилося можливим порівнювати результати розв’язання задачі PDPBS, отриманими методом, описаним в даній роботі, з результатами інших досліджень, не знімаючи при цьому обмеження, що враховуються тільки в даній роботі або тільки в інших дослідженнях.

Тому найбільш об’єктивним методом оцінки ефективності (тобто точності отриманого розв’язку та часу роботи алгоритму, що знадобився для отримання цього розв’язку) запропонованого в даній роботі евристичного методу розв’язання задачі PDPBS є отримання точних і евристичних розв’язків для одних і тих самих наборів вхідних даних (instances) за допомогою точного та евристичного алгоритмів, а потім порівняння вартості розв’язків та часу роботи цих алгоритмів.

Обчислювальні експерименти проводилися на персональному комп’ютері Dell Inspiron 5759, Intel Core i5-6200U (2.30GHz), 8 GM RAM.

4.3.3.1 Основні результати

У процессі проведення обчислювальних експериментів було отримано точні та евристичні розв’язки для різних наборів вхідних даних (instances).

Кожен набор вхідних даних являв собою комбінацію таких параметрів:

- однієї з 4 розмірностей задач: n = 3, 4, 5 та 6, тобто кількості пар pickup-delivery;
- однієї з 6 комбінацій лінійних розмірів вантажного відсіку та вантажопідйомності кожного транспортного засобу: вантажний відсік був кубом зі стороною W, H, L= 50, 70, 90, 110, 130, 150, а

вантажопідйомність відповідно дорівнювала Q = 100, 200, 300, 400, 500, 600;

- однієї з 5 комбінацій множин координат депо та пунктів pickup і delivery, а також розміру і ваги контейнерів. Координати пунктів pickup, delivery та депо були випадковим чином обрані з діапазону [1; 500000], а контейнери були кубами із довжиною ребра, випадково обраною з діапазону [1; 50] і мали вагу, також випадково обрану з діапазону [1; 100].

Для кожного з 4x6x5 = 120 наборів вхідних даних був отриманий точний розв'язок (тобто розв'язок для RBW = 100%) та 3 евристичних розв'язки для різних значень RBW = 10,30,50%. Для всіх наборів вхідних даних ймовірність перевірки тривимірних обмежень завантаження *check_prob* становила 0.2. Через складність процедури перевірки тривимірних обмежень завантаження, точні розв'язки було отримано тільки для невеликих розмірностей задач $n \leq 6$. Після отримання розв'язків, для кожного з них було порівняно вартість маршруту, отриманого за допомогою евристичного методу, з вартістю маршруту, отриманого за допомогою точного методу. Для цього було обчислено відносне збільшення вартості маршруту (оскільки вартість маршруту, отриманого за допомогою евристичного методу розв'язання, завжди більше або дорівнює вартості маршруту, отриманого за допомогою точного методу розв'язання):

$$\text{cost\_increase} = \frac{heu - ex}{ex},$$

де heu – вартість маршруту, отриманого за допомогою евристичного методу розв'язання; ex – вартість маршруту, отриманого за допомогою точного методу розв'язання.

Після отримання результатів, було проаналізовано:

1) залежність *cost_increase* від кількості пар pickup-delivery n та відносної ширини променю RBW в евристичному розв'язку (рис. 4.9);

2) залежність часу отримання точного (*RBW*=100%) та евристичних (*RBW*<100%) розв’язків від *n* та *RBW* (рис. 4.10);

3) відносну частоту виникнення ситуації, описаній у зауваженні *3.1*, для різних *n*, *RBW* і розмірів контейнерів (таблиці 4.4-4.7);

4) відносну частоту випадку, коли вартість маршруту, отриманого за допомогою евристичного методу, співпадає з вартістю маршруту, отриманого за допомогою точного методу (тобто ситуації, коли застосування евристики дозволило отримати оптимальний розв’язок задачі) для різних *n* і *RBW* (рис. 4.11).

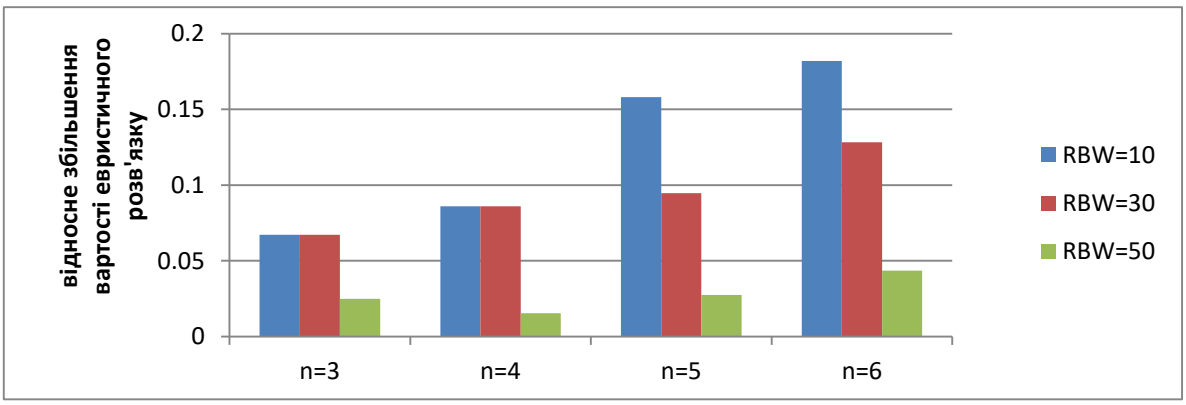


Рисунок 4.9. Залежність *cost_increase* (вісь *y*) від *n* та *RBW*

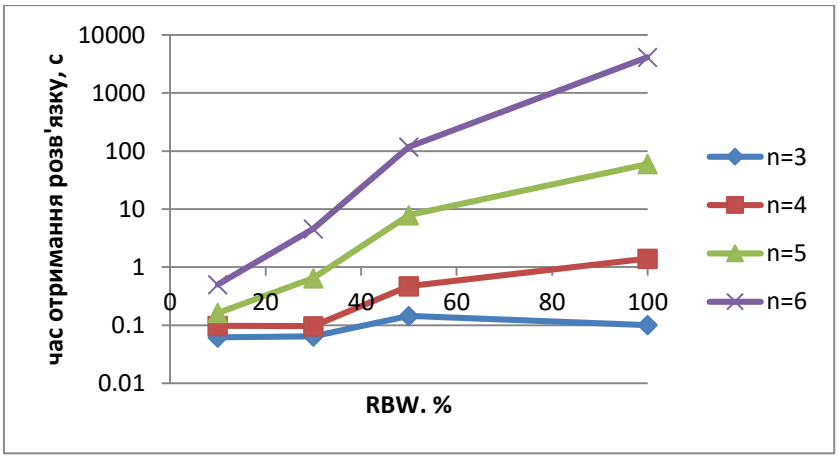


Рисунок 4.10. Час отримання евристичного (*RBW* < 100%) та точного (*RBW*=100%) розв’язків при різних значеннях кількості пар pickup-delivery (*n*) та відносної ширини променю (*RBW*)

Таблиця 4.4

Відносна частота виникнення ситуації, описаній у зауваженні 3.1, для різних *RBW* та довжин грані кубічного контейнера (*n* = 3)

	Довжина грані кубічного контейнера		
RBW, %	50	70	>70
10	0,12	0	0
30	0,12	0	0
50	0	0	0

Таблиця 4.5

Відносна частота виникнення ситуації, описаній у зауваженні 3.1, для різних *RBW* та довжин грані кубічного контейнера (*n* = 4)

	Довжина грані кубічного контейнера		
RBW, %	50	70	>70
10	0,76	0,2	0
30	0,7	0,2	0
50	0,23	0	0

Таблиця 4.6

Відносна частота виникнення ситуації, описаній у зауваженні 3.1, для різних *RBW* та довжин грані кубічного контейнера (*n* = 5)

	Довжина грані кубічного контейнера		
RBW, %	50	70	>70
10	0,58	0,2	0
30	0,29	0,2	0
50	0	0	0

Таблиця 4.7

Відносна частота виникнення ситуації, описаній у зауваженні 3.1, для різних *RBW* та довжин грані кубічного контейнера (*n* = 6)

	Довжина грані кубічного контейнера		
RBW, %	50	70	>70
10	0,47	0,4	0
30	0,11	0	0
50	0	0	0

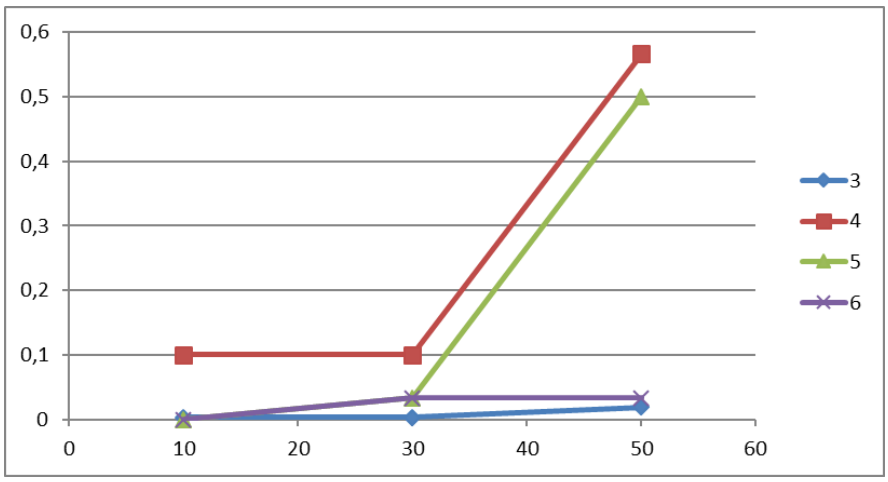


Рисунок 4.11. Відносна частота отримання оптимального розв’язання за допомогою евристики (вісь *y*) для різних *n* (*n* = 3, 4, 5, 6; див. легенду) і *RBW* (вісь *x*).

Аналіз показав, що запропонований метод розв'язання задачі PDP дозволяє отримати розв’язок на 1-4 порядки швидше, ніж повний перебір, водночас похибка становить 3-18%.

4.3.3.2 Додаткові результати

Також було проведено ще одну серію обчислювальних експериментів для аналізу того, як середній час роботи евристичного методу залежить від *check_prob*. Кожен набор вхідних даних є комбінацією деяких параметрів. В даному випадку, цими параметрами були:

- 4 розмірності задачі: *n* = 3, 4, 5 і 6;
- 3 варіанти тривимірних обмежень завантаження, де вантажний відсік транспортного засобу являв собою куб зі стороною *W*=*H*=*L*=50, 75, 150, а вантажопідйомність *Q* кожного транспортного засобу дорівнювала 100, 300, 600 відповідно;
- 3 комбінації множин координат депо та пунктів pickup і delivery, а також розміру і ваги контейнерів. Значення параметрів були отримані способом, описаним вище.

Для кожного з 4 x 3 x 3 = 36 наборів вхідних даних було отримано

наближені розв'язки для різних комбінацій *RBW* = 10,30,50% та *check_prob* = 0,1; 0,2; 0,3; 0,4. Отже, для кожного набору вхідних даних було отримано 3 x 4 = 12 евристичних розв'язків. Було проаналізовано, як час роботи евристичного алгоритму залежить від *check_prob* для різних *n* та *RBW*. Наведені нижче таблиці 4.8 - 4.11 містять час роботи алгоритму (в секундах), середній по 3 x 3 = 9 експериментам (3 варіанти тривимірних обмежень завантаження та 3 комбінації множин координат депо та пунктів pickup і delivery, а також розміру і ваги контейнерів) для одних і тих самих значень розмірності задачі *n* та відносної ширини променю *RBW*.

Таблиця 4.8

Залежність часу роботи евристичного алгоритму (в секундах) від *check_prob* та *RBW* (*n* = 3)

n=3	check_prob			
RBW	10	20	30	40
10	0.04	0.08	0.06	0.08
30	0.05	0.05	0.07	0.09
50	0.09	0.12	0.16	0.22

Таблиця 4.9

Залежність часу роботи евристичного алгоритму (в секундах) від *check_prob* та *RBW* (*n* = 4)

n=4	check_prob			
RBW	10	20	30	40
10	0.05	0.08	0.14	0.12
30	0.05	0.08	0.11	0.14
50	0.24	0.38	0.52	0.67

Таблиця 4.10

Залежність часу роботи евристичного алгоритму (в секундах) від *check_prob* та *RBW* (*n* = 5)

n=5	check_prob			
RBW	10	20	30	40
10	0.09	0.13	0.20	0.23
30	0.31	0.56	0.77	0.93
50	3.84	6.46	9.61	12.52

Таблиця 4.11

Залежність часу роботи евристичного алгоритму (в секундах) від *check_prob* та *RBW* (*n* = 6)

n=6	check_prob			
RBW	10	20	30	40
10	0.13	0.20	0.27	0.34
30	1.51	2.74	3.69	4.94
50	42.74	75.96	105.79	135.25

4.3.3.3 Експерименти на задачах великої розмірності

У даній серії експериментів було отримано евристичний розв'язок для задач великої розмірності ($n \le 50$) з різними *RBW*. Щоб прискорити час роботи алгоритму, вхідні дані були задані так, щоб тривимірні обмеження завантаження завжди задовольнялися (вантажопідйомність Q = 10000, а вантажний відсік транспортного засобу був кубом зі стороною 5000), отже їх перевірку було опущено. Кожен набір вхідних даних містить:

- одну 5 з комбінацій координат пунктів pickup-delivery та депо
- одну з 9 розмірностей задачі: n = 10, 15, 20, 25, 30, 35, 40, 45, 50.

Для кожного з 9 х 5 = 45 наборів вхідних даних було отримано 20 евристичних розв’язків для *RBW* = 0,25; 0,5 .... 5%.

Було встановлено обмеження на максимальний час роботи алгоритму в 1000 секунд. На наведеному нижче рис. 4.12 для різних розмірностей задачі показано максимальну величину *RBW*, для якої час роботи алгоритму отримання евристичного розв’язку не перевищував допустимі 1000 секунд. Ці результати можна розглядати як максимально точний розв’язок, який може бути отримано за встановлений час для кожного n.

Для розв’язків, що задовольняють обмеження часу, було проаналізовано залежність часу роботи (в секундах) алгоритму для отримання евристичного розв’язання від розмірності задачі n та відносної ширини променю *RBW* (рис. 4.13). Відмітимо, що вісь y має логарифмічний масштаб.

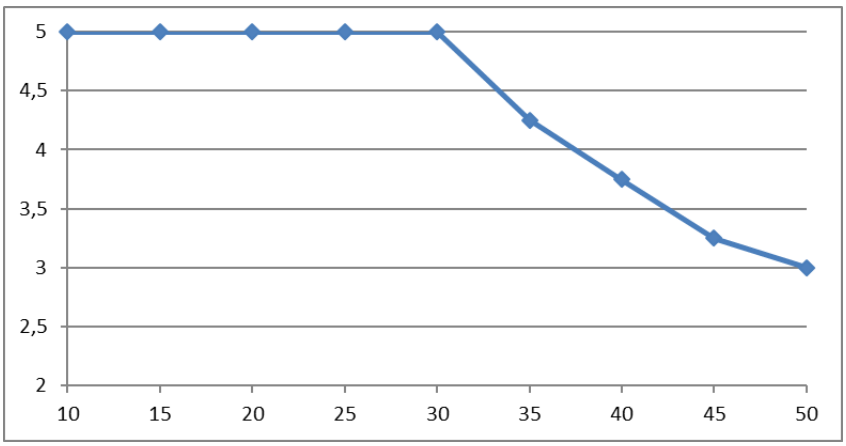


Рисунок 4.12. Максимальна відносна ширина променю (*RBW*, вісь *y*), за якої можна отримати евристичний розв'язок за час, менший 1000 секунд для різних розмірностей задачі *n* (вісь *x*).

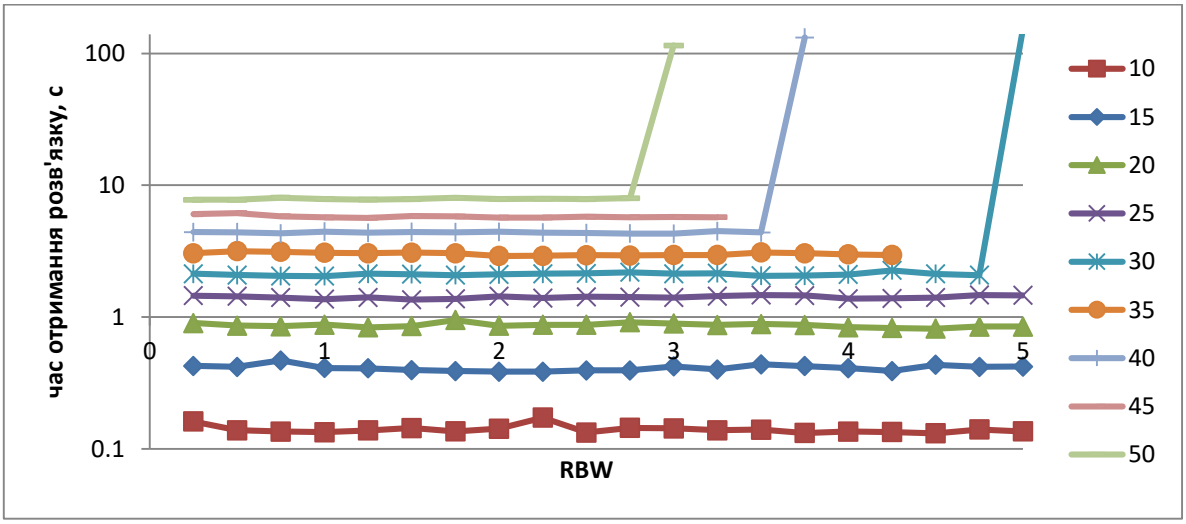


Рисунок 4.13. Залежність часу отримання евристичного розв'язку (в секундах, вісь *y*) від *n* (див. легенду) і *RBW* (вісь *x*)

4.4 Модуль розв'язання задачі TYSP2

Алгоритм розв'язання задачі складання розкладу руху вантажних поїздів з призначенням поїздів на залізничні колії (див. пункт 3.2) було реалізовано у вигляді програмного модуля на мові Python 2.7 та веб-фреймворку Flask.

Розроблений програмний модуль доступний онлайн за адресою: https://tsy-solver.herokuapp.com/ . Скріншоти програмного модуля наведено у додатку В.

Щоб побачити, як урахування призначення поїздів на колії впливає на якість розв'язання і час роботи алгоритму, було проведено обчислювальні експерименти з різноманітними варіантами вхідних даних. Кожен варіант вхідних даних є комбінацією наступних параметрів:

- одного з 9 варіантів кількості поїздів $N = 6,7, ..., 14$;
- одного з 4 варіантів кількості колій $G = 2,3,4,5$;
- одного з 4 варіантів вартості переміщення колія – колія:

$$c_{pq} = \text{cost\_coef} * |p - q| \text{ , cost\_coef} = 1,10,50,100;$$

- однієї з 5 різних кількостей цільових поїздів для кожного поїзду, коли кожний поїзд перевозить контейнери для 1,2,3,4 чи 5 інших поїздів. Позначимо цей параметр як *cargos_per_train*. Його можна виразити через розрідженість матриці A

$$\sum_{i=1}^{N} A_{ij} = 1,2,3,4,5, \quad j = 1,2...N.$$

Вагові коефіцієнти дорівнювали для всіх випадків:

$$\alpha_1 = \alpha_2 = \alpha_3 = 1.$$

Для цього на кожному з $9 \cdot 4 \cdot 4 \cdot 5 = 720$ варіантів вхідних даних задачу було розв'язано двічі: без призначення поїздів на колії (як це робилося в попередніх дослідженнях інших авторів, що розв'язували задачу на першому рівні деталізації [221]) та з призначенням (як запропоновано в даній роботі). Для кожної отриманої пари розв'язків розраховано відносне збільшення часу і відносне зниження вартості розв'язку з призначенням поїздів на залізничні колії порівняно з розв'язком без призначення

$$cost\_decrease = \frac{c_1 - c_2}{c_2}$$

$$time\_increase = \frac{t_2 - t_1}{t_1}$$

де c_1 і c_2 – вартість отриманного розв'яз, тобто значення цільової функції (3.20) для розв'язків без і з призначенням поїздів на залізничні колії, а t_1 і t_2 – час роботи алгоритму без і з призначенням поїздів на залізничні колії відповідно.

Обчислювальні експерименти проводилися на персональному комп'ютері Dell Inspiron 5759, Intel Core i5-6200U (2.30GHz), 8 GM RAM.

Детальну інформацію щодо вихідних даних та отриманих розв'язків можна знайти за адресою https://goo.gl/CqBjNe.

Результати аналізу впливу кожного з чотирьох вхідних параметрів на збільшення часу і зниження вартості наведено на рис. 4.14 – 4.17 (вісь *y* має логарифмічний масштаб).

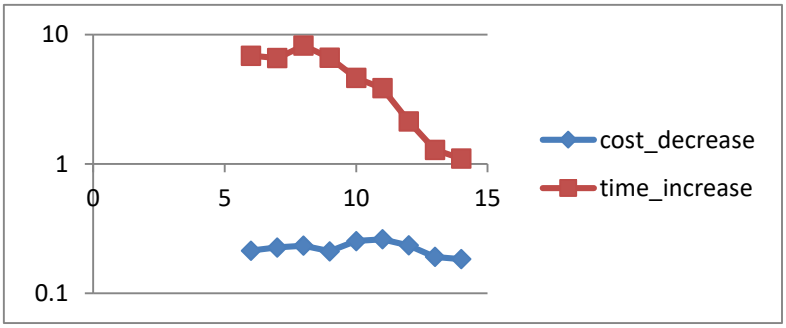


Рисунок 4.14. *cost_decrease* і *time_increase* (вісь *y*) для різних N (вісь *x*)

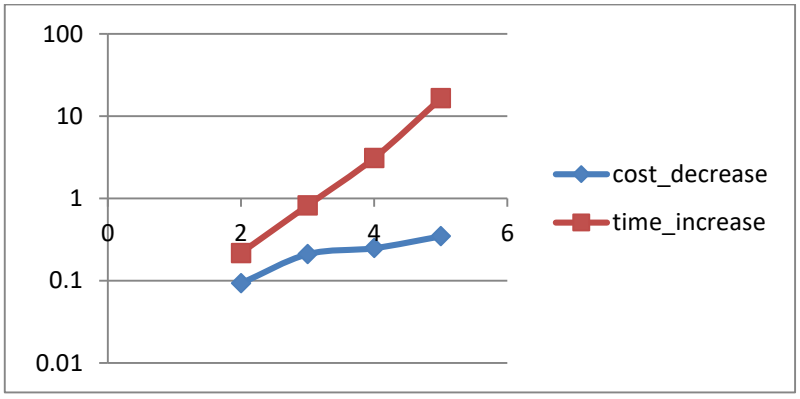


Рисунок 4.15. *cost_decrease* і *time_increase* (вісь *y*) для різних *G* (вісь *x*)

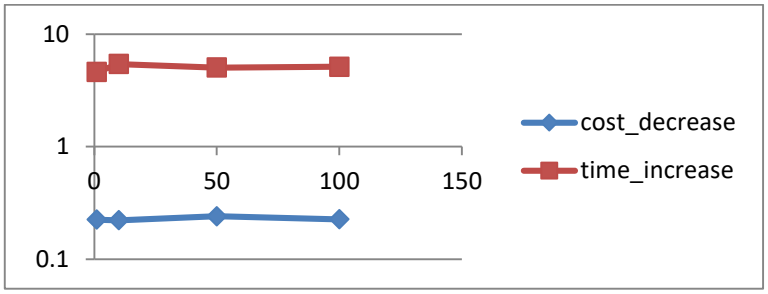


Рисунок 4.16. *cost_decrease* і *time_increase* (вісь *y*) для різних *cost_coef* (вісь *x*)

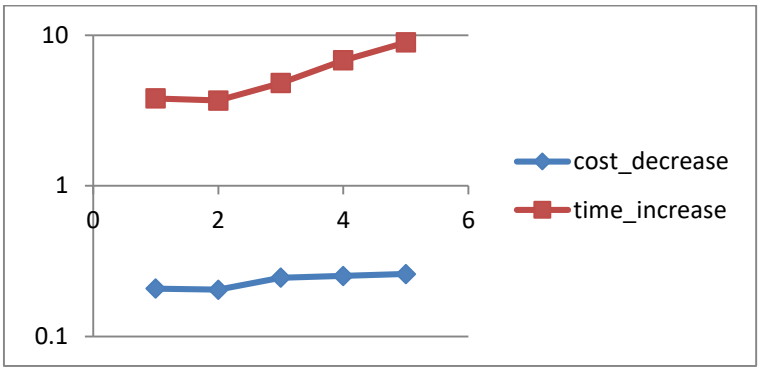


Рисунок 4.17. *cost_decrease* і *time_increase* (вісь *y*) для різних *cargos_per_train* (вісь *x*)

Графіки показують, що *G* і *cargos_per_train* безпосередньо впливають як на *cost_decrease*, так і на *time_increase*, *n* впливає в основному на *time_increase*, а *cost_coef* майже не впливає ані на *time_increase*, ані на *cost_decrease*.

4.5 Висновки по четвертому розділу

У даному розділі було описано розробку комплексу програм для розв'язання задач комбінаторної генерації та оптимізації, описаних у розділах 2 та 3. Наведено приклади роботи розроблених програмних модулів, описано обчислювальні експерименти, проведено аналіз результатів роботи алгоритмів та програмних модулів.

Описано розробку програмного модуля для генерації композиційних *k*-образів комбінаторних множин, було наведено приклад його роботи.

Розглянуто модуль генерації перестановок з частково заданою сигнатурою, що є модифікацією програмного модуля для генерації *k*-множин, описано приклад роботи розробленого модуля.

Для модуля розв’язання задачі PDPBS наведено приклад роботи, а також проведено оцінку точності результатів, отриманих за допомогою евристичного алгоритму. Аналіз результатів показав, що, незважаючи на складність розв’язання задачі PDPBS в поєднанні із задачею розміщення вантажів, отриманий в даній роботі евристичний метод дозволяє отримувати досить точні розв’язання за прийнятний час. При цьому описаний евристичний метод розв’язання задачі PDP має можливість балансування між точністю результатів та часом роботи.

Описано програмний модуль розв’язання задачі складання розкладу руху вантажних поїздів з урахуванням призначення поїздів на колії (TYSP2). Наведено скріншоти вхідних даних та результатів роботи розробленого програмного модуля. Розроблений модуль, як і модуль для розв’язання задачі PDPBS, доступний онлайн. Обчислювальні експерименти показали, що урахування призначення поїздів на колії дозволяє отримати кращі результати за рахунок збільшення обчислювальної складності.

Результати розділу опубліковано в роботах [40], [41], [57], [58].

ВИСНОВКИ

У дисертаційній роботі розвинуто існуючі стратегії та методи генерації комбінаторних конфігурацій та застосовано їх в математичному і комп’ютерному моделюванні та розв’язанні задач перевезення та обробки вантажів.

1. У роботі проаналізовано сучасний стан таких областей дослідження, як комбінаторна генерація та оптимізація. В результаті аналізу встановлено:

− розвиток стратегій та методів комбінаторної генерації, створення методів генерації *k*-множин є актуальною задачею;

− нерозв’язаними залишаються задачі перечислення та генерації перестановок з частково заданою сигнатурою, що є узагальненням перестановок з заданими підйомами та спусками;

− актуальним є дослідження задач перевезення та обробки вантажів, зокрема, задач вивозу та доставки зі спеціальними обмеженнями та задач складання розкладів руху вантажних поїздів та обробки вантажів на сортувальних станціях.

2. Набули подальшого розвитку стратегії та методи генерації комбінаторних конфігурацій.

3. Розроблено методи генерації *k*-множин.

4. Введено та досліджено перестановки з частково заданою сигнатурою.

5. Побудовано математичні моделі для задачі вивозу і доставки (Pickup and Delivery Problem) і задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції, що використовують комбінаторні конфігурації та враховують додаткові обмеження. Розроблено методи розв’язання зазначених класів задач на основі комбінаторної генерації, створено відповідні алгоритми та програмне забезпечення.

Результати обчислювальних експериментів показали, що запропонований метод розв'язання задачі PDP дозволяє отримати розв’язок на 1-4 порядки швидше, ніж повний перебір, водночас похибка становить 3-18%.

Врахування призначення поїздів на залізничні колії в задачі складання розкладу руху вантажних поїздів та обробки вантажів на сортувальній станції дозволяє знизити витрати на переміщення вантажів на 9-35%.

7. Отримані результати є теоретичною і практичною основою для розв'язання широких класів задач комбінаторної генерації та оптимізації. Зокрема:

− стратегія та узагальнений метод генерації комбінаторних конфігурацій можуть бути використані в методах розв'язання задач комбінаторної оптимізації, що використовують комбінаторну генерацію;

− метод та алгоритм генерації *k*-множин можуть бути використані для математичного моделювання багатьох задач, що мають складну структуру, зокрема, задачі упаковки *n*-вимірних паралелепіпедів, задачі Pickup and Delivery Problem та задачі складання розкладу вантажних поїздів на сортувальній станції;

− розроблені методи та алгоритми розв'язання задачі Pickup and Delivery Problem можуть бути використані в діяльності транспортних компаній для підвищення ефективності та зниження вартості доставки вантажів;

− метод та алгоритм розв'язання задачі складання розкладу вантажних поїздів можуть бути безпосередньо використані для оптимізації роботи сортувальних станцій, а урахування призначення поїздів на колії – для зниження витрат на роботу портальних кранів.

8. Практичне значення результатів роботи підтверджується їх впровадженням в держбюджетні наукові дослідження та у виробництво, про що свідчать відповідні акти.

9. Можливі напрями продовження досліджень за тематикою дисертації:

− розробка методів та алгоритмів генерації *k*-множин, а також перестановок з частково заданою сигнатурою, що мають меншу обчислювальну складність порівняно із запропонованими у даній роботі;

- використання більш ефективних методів та алгоритмів кластеризації для розв’язання верхнього рівня задачі вивозу та доставки (Pickup and Delivery Problem);
- розробка більш ефективних методів та алгоритмів для упаковки контейнерів та побудови маршрутів транспортних засобів для розв’язання задачі Pickup and Delivery Problem на нижньому рівні;
- створення більш ефективних методів отримання оптимального призначення поїздів на колії в задачі складання розкладу вантажних поїздів на сортувальній станції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] N. Christofides. Combinatorial optimization. New York: Wiley. 1979.

[2] A. Schrijver. Combinatorial optimization: polyhedra and efficiency. Springer. 2003.

[3] C. H. Papadimitriou, K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. 2013.

[4] P. Toth, D. Vigo. Vehicle routing: problems, methods, and applications. Society for Industrial and Applied Mathematics. 2014.

[5] P. Panos, M. Pardalos, D. Du. Handbook of combinatorial optimization. 2013.

[6] M. Iori, S. Martello. Routing problems with loading constraints. TOP. 2010. Vol. 18, No. 1. pp. 4–27.

[7] Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации. Киев: Наукова Думка. 1988.

[8] М. З. Згуровский, А. А. Павлов. Труднорешаемые задачи комбинаторной оптимизации в планировании и принятии решений. Киев: Наукова думка. 2016.

[9] Ю. Стоян, С. Яковлев. Математические модели и оптимизационные методы геометрического проектирования. Киев: Наукова думка. 1986.

[10] Ю. Г. Стоян, С. В. Яковлев, О. С. Пичугина. Эвклидовы комбинаторные конфигурации: монография. Харьков: Константа. 2017.

[11] Семенова Н.В., Колєчкіна Л.М. Векторні задачі дискретної оптимізації на комбінаторних множинах. Київ: Наукова думка. 2009.

[12] С. В. Яковлев, Н. И. Гиль, В.М. Комяк, И. В. Аристова. Элементы теории геометрического проектирования. Киев: Наукова думка. 1995.

[13] Y. Stoyan, A. Pankratov, T. Romanova. Placement Problems for Irregular Objects: Mathematical Modeling, Optimization and Applications. Springer, Cham. 2017. pp. 521–559.

[14] Ю. Г. Стоян, О. О. Ємець. Теорія і методи евклідової комбінаторної оптимізації. Інститут системних досліджень освіти. Київ. 1993.

[15] И. В. Гребенник. Математические модели и методы комбинаторной оптимизации в геометрическом проектировании. Харьковский национальный университет радиоэлектроники. 2006.

[16] T. Crainic, K. Kim. Intermodal transportation. *Handbooks in operations research and management science*. 2007. Vol. 14.

[17] P. Arnold, D. Peeters, I. Thomas. Modelling a rail/road intermodal transportation system. *Transportation Research Part E: Logistics and Transportation Review*. 2004. Vol. 40, No. 3. pp. 255–270.

[18] B. Slack. Intermodal Transportation. 2008. 204 p.

[19] T. Bartók, C. Imreh. Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints. *Acta Cybernetica*. 2011. Vol. 20, No. 1. pp. 17–33.

[20] D. E. Mulcahy. Warehouse distribution and operations handbook. McGraw-Hill. 1994.

[21] J. Perl, M. S. Daskin. A warehouse location-routing problem. *Transportation Research Part B: Methodological*. 1985. Vol. 19, No. 5. pp. 381–396.

[22] C. Caballini, S. Sacone, M. Saeednia. Cooperation among truck carriers in seaport containerized transportation. *Transportation Research Part E: Logistics and Transportation Review*. 2016. Vol. 93. pp. 210-216.

[23] E. Kozan. Optimising container transfers at multimodal terminals. *Mathematical and Computer Modelling*. 2000. Vol. 31, No. 10–12. pp. 235–243.

[24] K. H. Kim and H. O. Günther. Container terminals and cargo systems: Design, operations management, and logistics control issues. Berlin, Heidelberg: Springer. 2007.

[25] D. Knuth. The Art of Computer Programming. Volume 4, Generating All Tuples and Permutations. Fascicle 2. Addison-Wesley. 2005.

[26] D. Knuth. The Art of Computer Programming. Volume 4: Generating all Combinations and Partitions. Fascicle 3. Addison-Wesley. 2005.

[27] F. Ruskey. Combinatorial generation. 2003.

[28] D. L. Kreher and D. R. Stinson. Combinatorial Algorithms: Generation, Enumeration, and Search. CRC press. 1998.

[29] A. Tucker. Applied combinatorics. John Wiley and Sons. 2012.

[30] F. Roberts, B. Tesman. Applied Combinatorics, Second Edition. Chapman and Hall/CRC. 2009.

[31] M. Keller, W. Trotter. Applied Combinatorics. 2016.

[32] M. Lothaire. Applied Combinatorics on Words. Cambridge: Cambridge University Press. 2005.

[33] S. Kumar, R. Panneerselvam. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*. 2012. Vol. 4, No. 3. pp. 66-75.

[34] M. W. P. Savelsbergh, M. Sol. The General Pickup and Delivery Problem. *Transportation Science*. 1995. Vol. 29, No. 1. pp. 17–29.

[35] И. Гребенник, А. Панкратов, А. Чугай, А. Баранов. Упаковка n-мерных параллелепипедов с возможностью изменения их ортогональной ориентации в n-мерном параллелепипеде. *Кибернетика и системный анализ*. 2010. № 46(5). С. 122–131.

[36] И. Гребенник. Классы композиционных образов комбинаторных множеств в математических моделях задач геометрического проектирования. *Радиоэлектроника и информатика*. 2005. №3.

[37] И. Гребенник, А. Баранов. Оценки минимума выпуклых функции на классах комбинаторных множеств перестановок. *Радіоелектроніка, інформатика*. 2009. №1(20).

[38] И. Гребенник, А. Баранов. Математическое моделирование и решение некоторых многокритериальных задач размещения параллелепипедов. *Збірник наукових праць Харківського університету Повітряних сил*. 2010. № 2. С. 49-55.

[39] О. С. Литвиненко. Комбінаторні методи вибору оптимальних структур багаторівневих систем управління і автоматики. ХНУРЕ, Кафедра Системотехніки. Харків. 2014. 99 с.

[40] R. Dupas, I. Grebennik, O. Lytvynenko, O. Baranov. An Heuristic Approach to Solving the one-to-one Pickup and Delivery Problem with Three-dimensional Loading Constraints. *International Journal of Information Technology and Computer Science Information Technology and Computer Science*. 2017. Vol. 10, No. 10. pp. 1–12.

[41] I. Grebennik, R. Dupas, O. Lytvynenko, I. Urniaieva. Scheduling freight trains in rail-rail transshipment yards with train arrangements. *International Journal of Intelligent Systems and Applications*. 2017. Vol. 9, No. 10.

[42] Ю. Г. Стоян, И. В. Гребенник. Композиционные образы комбинаторных множеств и некоторые их свойства. *Пробл. машиностроения*. 2005. № 8(3). С. 56–62.

[43] Ю. Г. Стоян, И. В. Гребенник. Описание классов комбинаторных конфигураций на основе отображений. *Доповіді НАН України*. 2008. № 10. С. 28-31.

[44] Y. Stoyan, I. Grebennik. Description and Generation of Combinatorial Sets Having Special Characteristics. *Biomedical fuzzy and human sciences: the official journal of the Biomedical Fuzzy Systems Association*. 2013. Vol. 18, No. 1. pp. 83–88.

[45] И. Гребенник. Описание и генерация перестановок, содержащих циклы. *Кибернетика и системный анализ*. 2010. № 6. С. 97-105.

[46] И. В. Гребенник, А. С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. *Кибернетика и системный анализ*. 2012. № 48(6). С. 96–105.

[47] I. Grebennik, O. Lytvynenko. Random generation of combinatorial sets with special properties. *ECONTECHMOD : An International Quarterly Journal on Economics of Technology and Modelling Processes*. 2016. Vol. 5, No. 4. pp. 43–48.

[48] I. Grebennik, O. Lytvynenko. Random and exhaustive generation of combinatorial sets with special properties. *5th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2014)*. 2014.

[49] О. С. Литвиненко, И. В. Гребенник. Генерация комбинаторных множеств с заданными свойствами. *XIX International Conference 'Problems of decision making under uncertainties'*. 2012.

[50] И. В. Гребенник, А. С. Литвиненко. Генерация перестановок с частично определенным порядком следования элементов. *Бионика интеллекта*. 2017. № 2(89). С. 26–30.

[51] Y. G. Stoyan, I. Grebennik, V. Kalashnikov, O. Lytvynenko. Enumeration and Generation of Permutations with a Partially Fixed Order of Elements. *International Journal of Combinatorial Optimization Problems and Informatics*. 2017. Vol. 8, No. 1. pp. 19–30.

[52] И. В. Гребенник, А. С. Литвиненко. Перечисление и генерация перестановок с частично заданным порядком следования элементов. *Сучасні проблеми машинобудування*. 2016.

[53] R. Dupas, I. Grebennik, O. Lytvynenko. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. *V Konferencja Zastosowań Matematyki w Technice, Informatyce i Ekonomii*. 2015.

[54] R. Dupas, I. Grebennik, O. Lytvynenko. Combinatorial mathematical model and decision strategy for one-to-one pickup and delivery problem with 3D loading constraints. *Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT-2015*. 2015.

[55] O. Lytvynenko, O. Baranov, R. Dupas, I. Grebennik. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. *ILS 2016 - 6th International Conference on Information Systems, Logistics and Supply Chain*. 2016.

[56] I. Grebennik, R. Dupas, O. Lytvynenko, I. Urniaieva. Deciding on train arrangement in freight trains scheduling problem. *7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)*. 2017.

[57] I. Grebennik, R. Dupas, O. S. Lytvynenko, O. Baranov, I. Urniaieva. Developing software for solving some combinatorial generation and optimization problems. *7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)*. 2017.

[58] И. В. Гребенник, О. С. Титова, А. С. Литвиненко. Оптимизация линейной функции на множестве циклических перестановок. *Бионика интеллекта*. 2012. № 2 (67).

[59] I. V. Grebennik, O. S. Lytvynenko, O. S. Titova. Optimization of linear functions on cyclic permutations. *Proc. XX International Conf. 'Problems of decision making under uncertainties'*. 2012. pp. 43–44.

[60] Б. Я. Советов, С. В. Яковлев. Моделирование систем. Москва: Издательство Юрайт. 2012.

[61] О. Д. Шарапов, В. Д. Дербенцев. Економічна кібернетика. Київ: КНЕУ. 2004.

[62] C. Berge. Principes de combinatoire. Paris: Dunod. 1968.

[63] В. Н. Сачков. Комбинаторные методы дискретной математики. Москва: Наука. 1975.

[64] С. В. Яковлев. Свойства задач комбинаторной оптимизации на полиэдрально-сферических множествах. *Кибернетика и системный анализ*. 2018. № 54(1). С. 111–123.

[65] R. P. Stanley. Enumerative combinatorics. Belmont, CA: Wadsworth Publ. 1986.

[66] M. Bóna. Handbook of enumerative combinatorics. CRC Press. 2015.

[67] T. Feder, R. Motwani. Clique Partitions, Graph Compression and Speeding-Up Algorithms. *Journal of Computer and System Sciences*. 1995. Vol. 51, No. 2. pp. 261–272.

[68] T. Feder, R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *Proceedings of the twenty-third annual ACM symposium on Theory of computing - STOC '91*. 1991. pp. 123–133.

[69] C. Bron, J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*. 1973. Vol. 16, No. 9. pp. 575–577.

[70] И. Сергиенко, О. Емец, А. Емец. Задачи оптимизации с интервальной неопределенностью: метод ветвей и границ. *Кибернетика и системный*. 2013. № 49(5). С. 38–50.

[71] О. Емец. Евклидовы комбинаторные множества и оптимизация на них. Новое в математическом программировании. 1992.

[72] О. Емец, А. Роскладка. О комбинаторной оптимизации в условиях неопределенности. *Кибернетика и системный анализ*. 2008. № 44(5). С. 35–44.

[73] О. Емец, Н. Романова. Оптимизация на полиперестановках. Монография. 2010.

[74] О. Емец, Т. Барболина. Комбинаторная оптимизация на размещениях. Монография. 2008.

[75] Семенова Н.В., Колечкина Л.М. Многокритериальные задачи на комбинаторном множестве полиразмещений: полиэдральный подход к их решению. *Кибернетика и системный анализ*. 2009. Vol. 3. С. 118-126.

[76] Н. Семенова, Л. Колечкина. Векторные задачи оптимизации с линейными критериями на нечетко заданном комбинаторном множестве альтернатив. *Кибернетика и системный анализ*. 2011. № 47(2). С. 88–99.

[77] Н. Семенова. Условия оптимальности в векторных задачах комбинаторной оптимизации. *Теорія оптимальних рішень*. 2008. № 7. С. 153-160.

[78] Л. Гуляницкий, И. Сергиенко. Метаэвристический метод деформированного многогранника в комбинаторной оптимизации. *Кибернетика и системный анализ*. 2007. № 6. С. 70-79.

[79] И. Сергиенко, Т. Лебедева, В. Рощин. Приближенные методы решения дискретных задач оптимизации. Киев: Наукова Думка. 1980.

[80] Л. Гуляницкий. Модифицированные алгоритмы вероятностного моделирования в комбинаторной оптимизации. Технология и методы решения задач прикладной математики. Ин-т кибернетики им. ВМ Глушкова АН Украины, Киев. 1991. С. 10-14.

[81] И. Сергиенко, Л. Гуляницкий. Классификация прикладных методов комбинаторной оптимизации. *Кибернетика и системный анализ*. 2009. № 5. С. 71-83.

[82] Сергиенко И.В., Шило В.П.. Задачи дискретной оптимизации. Проблемы, методы исследования, решения. Монография. Нац. акад. наук Украины, Ин-т кибернетики им. В. М. Глушкова. Киев. 2003.

[83] I. V. Sergienko. Methods of optimization and systems analysis for problems of transcomputational complexity. New York: Springer. 2012.

[84] Л. Гуляницький, О. Ю. Мулеса, Прикладні методи комбінаторної оптимізації: навчальний посібник. Київ: Видавничо-поліграфічний центр Київський університет. 2016.

[85] S. V. Yakovlev, O. A. Valuiskaya. Optimization of Linear Functions at the Vertices of a Permutation Polyhedron with Additional Linear Constraints. *Ukrainian Mathematical Journal*. 2001. Vol. 53, No. 9. pp. 1535–1545.

[86] С. В. Яковлев, И. В. Гребенник. О некоторых классах задач оптимизации на комбинаторных множествах размещений. Известия высших учебных заведений. Математика. 1991. № 11. С. 1991.

[87] С. В. Яковлев. О комбинаторной структуре задач оптимального размещения геометрических объектов. Доклады НАН Украины. 2017. № 9. С. 63-68.

[88] Г. П. Донець, Колєчкіна Л.М. Екстремальні задачі на комбінаторних конфігураціях. Полтава: ПУЕТ. 2001.

[89] Сергиенко И.В., Шило В.П. Современные подходы к решению сложных задач дискретной оптимизации. Проблемы управления и информатики. 2016. № 1. С. 32-40.

[90] В. А. Емеличев, М. М. Ковалев, М. К. Кравцов. Многогранники, графы, оптимизация (комбинаторная теория многогранников). Москва: Наука. 1981.

[91] W. Cook. Combinatorial optimization. Wiley. 1998.

[92] B. H. Korte and J. Vygen. Combinatorial optimization: theory and algorithms. New York: Springer. 2012.

[93] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. The Traveling Salesman Problem : a Computational Study. Princeton University Press. 2011.

[94] G. Gutin, A. P. Punnen. The Traveling Salesman Problem and Its Variations. Vol. 12. Boston, MA: Springer US. 2007.

[95] В. Михалевич, А. Кукса. Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов. Наука, Москва. 1983. 208 с.

[96] T. Gonzalez. Handbook of approximation algorithms and metaheuristics. CRC Press. 2007.

[97] Ю. Кочетов. Вычислительные возможности локального поиска в комбинаторной оптимизации. *Журнал вычислительной математики и математической физики*. 2008. № 45(8). С. 778–807.

[98] P. Hansen, N. Mladenović, J. A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*. 2010. Vol. 175, No. 1. pp. 367–407.

[99] S. Kirkpatrick, C. Gelatt, M. Vecchi. Optimization by simulated annealing. *Science*. 1983. Vol. 220, No. 4598. pp. 671–680.

[100] Л. Гуляницкий. Решение задач комбинаторной оптимизации алгоритмами ускоренного вероятностного моделирования. *Компьютерная математика*. 2004. №1.

[101] F. Glover, M. Laguna. Tabu Search. Handbook of Combinatorial Optimization, New York, NY: Springer New York. 2013. pp. 3261–3362.

[102] K. De Jong. Evolutionary computation: a unified approach. MIT Press. 2006.

[103] P. Moscato, C. Cotta. A Gentle Introduction to Memetic Algorithms. Handbook of Metaheuristics. Boston: Kluwer Academic Publishers. 2003. pp. 105–144.

[104] M. Dorigo, C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*. 2005. Vol. 344, No. 2–3. pp. 243–278.

[105] V. Maniezzo, A. Carbonaro. Ant Colony Optimization: An Overview. Springer, Boston, MA. 2002. pp. 469–492.

[106] В. А. Емеличев, М. М. Ковалев, М. К. Кравцов, Многогранники. Графы. Оптимизация. Наука. 1981.

[107] Н. В. Семенова, Л. М. Колечкина, А. Н. Нагорная. Подход к решению векторных задач дискретной оптимизации на комбинаторном множестве перестановок. *Кибернетика и системный анализ*. 2008. № 44(3). С. 158–172.

[108] О. С. Пичугина, С. В. Яковлев. Непрерывные функциональные представления в задачах дискретной оптимизации: монография. Харьков: Золотая миля. 2018.

[109] A. Nijenhuis, H. S. Wilf. Combinatorial algorithms for computers and calculators. Academic Press. 1978.

[110] В. Липский. Комбинаторика для программистов. Москва: Мир. 1988.

[111] В. В. Кручинин. Методы построения алгоритмов генерации и нумерации комбинаторных объектов на основе деревьев И/ИЛИ. Томск: В-Спектр. 2007.

[112] Б. Я. Рябко. Быстрая нумерация комбинаторных объектов. *Дискретная математика*. 1998. № 10(2). С. 101–119.

[113] В. Амелькин, Ю. Дробышев. Методы нумерационного кодирования. Новосибирск: Наука. 1986.

[114] G. Pólya, R. C. Read. Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds. New York, NY: Springer New York. 1987.

[115] Кинг Р. Химические приложения топологии и теории графов. Москва: Мир. 1987.

[116] Н. Колчанов, В. Суслов, К. Гунбин. Моделирование биологической эволюции: регуляторные генетические системы и кодирование сложности биологической организации. *Вавиловский журнал генетики и селекции*. 2004. № 8(29). С. 86–99.

[117] Д. Гасфилд. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. Санкт-Петербург: Невский Диалект. 2003.

[118] Э. Рейнгольд, Ю. Нивергельт, Н. Део. Комбинаторные алгоритмы. Теория и практика. М.: Мир. 1980.

[119] E. Barcucci, A. Del Lungo, E. Pergola, R. Pinzani. ECO:a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*. 1999. Vol. 5, No. 4–5. pp. 435–490.

[120] S. Bacchelli, E. Barcucci, E. Grazzini, E. Pergola. Exhaustive generation of combinatorial objects by ECO. *Acta Informatica*. 2004. Vol. 40, No. 8. pp. 585–602.

[121] C. Banderier, M. Bousquet-Mélou, A. Denise, P. Flajolet, D. Gardy, D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Mathematics*. 2002. Vol. 246, No. 1–3. pp. 29–55.

[122] A. Del Lungo, E. Duchi, A. Frosini. On the generation and enumeration of some classes of convex polyominoes. *The electronic journal of combinatorics*. 2004. Vol. 11. No. 1.

[123] P. Flajolet, P. Zimmermann, B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*. 1994. Vol. 132, No. 1–2. pp. 1–35.

[124] C. Martínez, X. Molinero. A generic approach for the unranking of labeled combinatorial classes. *Random Structures & Algorithms*. 2001. Vol. 19, No. 3–4. pp. 472–497.

[125] X. Molinero, O. Serra. Ordered generation of classes of combinatorial structures. University Politecnical of Catalunya. 2005.

[126] C. Martınez, X. Molinero. Generic algorithms for the generation of combinatorial objects. Mathematical Foundations of Computer Science. 2003. pp. 572–581.

[127] В. Н. Потапов. Обзор методов неискажающего кодирования дискретных источников. *Дискретный анализ и исследование операций*. 1999. № 6(4). С. 49–91.

[128] Ю. Стоян. Об одном обобщении функции плотного размещения. *Докл. АН УССР. Сер. А*. 1980. Vol. 8. С. 1980.

[129] Ю. Г. Стоян. Ф-функция n-мерных параллелепипедов. *Доповіді НАН України*. 2005. №. 3. С. 22-27.

[130] Y. G. Stoyan. Ф-function and its basic properties. *Доп. НАН України*. 2001. Vol. 8. С. 112-117.

[131] Ю. Стоян, Т. Романова, Г. Шайтхауэр. Математическое моделирование взаимодействий базовых геометрических 3D объектов. *Кибернетика и системный анализ*. 2005. № 41(3). С. 19–31.

[132] I. Niven. A combinatorial problem of finite sequences. *Nieuw Arch. Wisk.* 1968. Vol. 16, No. 3. pp. 116–123.

[133] M. Abramson. A simple solution of Simon Newcomb's problem. *Journal of Combinatorial Theory, Series A.* 1975. Vol. 18, No. 2. pp. 223–225.

[134] N. G. De Bruijn. Permutations with given ups and downs. *Nieuw Arch. Wisk.* 1970. Vol. 18, No. 3. pp. 61–65.

[135] H. O. Foulkes. Enumeration of permutations with prescribed up-down and inversion sequences. *Discrete Mathematics*. 1976. Vol. 15, No. 3. pp. 235–252.

[136] G. Viennot. Permutations ayant une forme donnée. *Discrete Mathematics*. 1979. Vol. 26, No. 3. pp. 279–284.

[137] D. R. van Baronaigien, F. Ruskey. Generating permutations with given ups and downs. *Discrete applied mathematics*. 1992. Vol. 36, No. 1. pp. 57–65.

[138] R. P. Stanley. A survey of alternating permutations. *Contemp. Math*. 2010. Vol. 531. pp. 2010.

[139] B. Bauslaugh, F. Ruskey. Generating alternating permutations lexicographically. *BIT Numerical Mathematics*. 1990. Vol. 31, No. 1. pp. 17–26.

[140] T. Gannon. The cyclic structure of unimodal permutations. *Discrete Mathematics*. 2001. Vol. 237, No. 1–3. pp. 149–161.

[141] J.-Y. Thibon. The Cycle Enumerator of Unimodal Permutations. *Annals of Combinatorics*. 2001. Vol. 5, No. 3–4. pp. 493–500.

[142] S. Elizalde, Y. Roichman. Arc permutations. *Journal of Algebraic Combinatorics*. 2014. Vol. 39, No. 2. pp. 301–334.

[143] Y. Takenouchi. 'Digital' Entropy Induced by Unimodal Cyclic Permutations. *Bulletin of the Malaysian Mathematical Sciences Society*. Feb. 2016. pp. 1–31.

[144] D. Pisinger, S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*. 2007. Vol. 34, No. 8. pp. 2403–2435.

[145] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*. 2013. Vol. 225, No. 1. pp. 1–11.

[146] B. Golden, S. Raghavan, E. Wasil. The vehicle routing problem: latest advances and new challenges. Springer Science & Business Media. 2008.

[147] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*. 1992. Vol. 59, No. 3. pp. 345–358.

[148] G. Laporte, M. Gendreau, J.-Y. Potvin, F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*. 2000. Vol. 7, No. 4–5. pp. 285–300.

[149] P. Toth, D. Vigo. Vehicle routing: problems, methods, and applications. Society for Industrial and Applied Mathematics. 2001.

[150] А. В. Панишев. Модели и методы оптимизации замкнутых маршрутов на транспортной сети: монография. ЖГТУ, Житомир. 2014. 316 с.

[151] P. Toth, D. Vigo. Branch-and-bound algorithms for the capacitated VRP. The vehicle routing problem. Society for Industrial and Applied Mathematics, pp. 29–51. 2001.

[152] M. Gendreau, G. Laporte, J. Potvin. Metaheuristics for the capacitated VRP. The vehicle routing problem. Society for Industrial and Applied Mathematics, pp. 129–154. 2001.

[153] D. Naddef, G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. The vehicle routing problem. Society for Industrial and Applied Mathematics, pp. 53–84. 2001.

[154] P. Toth, D. Vigo, J. Desrosiers, M. M. Solomon, F. Soumis. VRP with Time Windows. The vehicle routing problem. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. 2001. pp. 157–193.

[155] B. Kallehauge, J. Larsen, O. B. G. Madsen, M. M. Solomon. Vehicle Routing Problem with Time Windows. Column Generation, New York: Springer-Verlag. 2005. pp. 67–98.

[156] M. W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*. 1985. Vol. 4, No. 1. pp. 285–305.

[157] N. A. El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science*. 2010. Vol. 22, No. 3. pp. 123–131.

[158] S. Belhaiza, R. M'Hallah, G. Ben Brahim. A new Hybrid Genetic Variable Neighborhood search heuristic for the Vehicle Routing Problem with Multiple Time Windows. _2017 IEEE Congress on Evolutionary Computation (CEC). 2017. pp. 1319–1326.

[159] I. Chao, B. Golden, E. Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*. 1993. Vol. 13, No. 3–4. pp. 371–406.

[160] W. Ho, G. Ho, P. Ji, H. Lau. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*. 2008. Vol. 21, No. 4. pp. 548–557.

[161] J. Cordeau, M. Gendreau, G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problem. *Computers & Operations Researc*. 1996. Vol. 23, No. 3. pp. 229–235.

[162] M. Jin, K. Liu, B. Eksioglu. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*. 2008. Vol. 36, No. 2. pp. 265–270.

[163] C. Archetti, M. G. Speranza. The Split Delivery Vehicle Routing Problem: A Survey. The Vehicle Routing Problem: Latest Advances and New Challenges, Boston, MA: Springer US. 2008. pp. 103–122.

[164] J. Belenguer, M. Martinez, E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*. 2000. Vol. 48, No. 5. pp. 801–810.

[165] C. Archetti, M. Speranza, A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation science*. 2006. Vol. 40, No. 1. pp. 64–73.

[166] S. Chen, B. Golden, E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*. 2007. Vol. 49, No. 4. pp. 318–329.

[167] B. Yao, P. Hu, M. Zhang, S. Wang. Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem. *SIMULATION*. 2013. Vol. 89, No. 6. pp. 762–770.

[168] P. Francis, K. Smilowitz. Modeling techniques for periodic vehicle routing problems. *Transportation Research Part B: Methodological*. 2006. Vol. 40, No. 10. pp. 872–884.

[169] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*. 2012. Vol. 60, No. 3. pp. 611–624.

[170] M. Gaudioso, G. Paletta. A Heuristic for the Periodic Vehicle Routing Problem. *Transportation Science*. 1992. Vol. 26, No. 2. pp. 86–92.

[171] E. Angelelli, M. Grazia Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*. 2002. Vol. 137, No. 2. pp. 233–247.

[172] K. Chepuri, T. Homem-de-Mello. Solving the Vehicle Routing Problem with Stochastic Demands using the Cross-Entropy Method. *Annals of Operations Research*. 2005. Vol. 134, No. 1. pp. 153–181.

[173] K. C. Tan, C. Y. Cheong, C. K. Goh. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. *European Journal of Operational Research*. 2007. Vol. 177, No. 2. pp. 813–839.

[174] L. M. Hvattum, A. Løkketangen, G. Laporte. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. *Transportation Science*. 2006. Vol. 40, No. 4. pp. 421–438.

[175] W.-H. Yang, K. Mathur, R. H. Ballou. Stochastic Vehicle Routing Problem with Restocking. *Transportation Science*. 2000. Vol. 34, No. 1. pp. 99–112.

[176] M. Gendreau, O. Jabali, W. Rei. Stochastic vehicle routing problems. Vehicle routing: problems, methods, and applications. 2014. pp. 213–239.

[177] M. Gendreau, G. Laporte, R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*. 1996. Vol. 88, No. 1. pp. 3–12.

[178] J. Bard, L. Huang, M. Dror, P. Jaillet. A branch and cut algorithm for the VRP with satellite facilities. *IIE transactions*. 1998. Vol. 30, No. 9. pp. 821–834.

[179] J. Bard, L. Huang, P. Jaillet, M. Dror. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation science*. 1998. Vol. 32, No. 2. pp. 189–203.

[180] S. N. Parragh, K. F. Doerner, R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*. 2008. Vol. 58, No. 1. pp. 21–51.

[181] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*. 2007. Vol. 15, No. 1. pp. 1–31.

[182] J. Cordeau, G. Laporte, S. Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. The vehicle routing problem: latest advances and new challenges. Vol. 43. 2008. pp. 327–357.

[183] J. Cordeau, G. Laporte. The dial-a-ride problem: models and algorithms. Annals of operations research. Vol. 153. No. 1, p. 29. 2007.

[184] J.-F. Cordeau, G. Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*. 2003. Vol. 1, No. 2. pp. 89–101.

[185] S. Mitrović-Minić and Snezana. The dynamic pickup and delivery problem with time windows. Proquest Information and Learning. 2001.

[186] G. Berbeglia, J.-F. Cordeau, G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*. 2010. Vol. 202, No. 1. pp. 8–15.

[187] Y. Dumas, J. Desrosiers, F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*. 1991. Vol. 54, No. 1. pp. 7–22.

[188] S. Ropke, D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*. 2006. Vol. 40, No. 4. pp. 455–472.

[189] S. Ropke, J.-F. Cordeau. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*. 2009. Vol. 43, No. 3. pp. 267–286.

[190] R. Baldacci, E. Bartolini, A. Mingozzi. An Exact Algorithm for the Pickup and Delivery Problem with Time Windows. *Operations Research*. 2011. Vol. 59, No. 2. pp. 414–426.

[191] R. Liu, X. Xie, V. Augusto, C. Rodriguez. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*. 2013. Vol. 230, No. 3. pp. 475–486.

[192] M. Christiansen, K. Fagerholt. The Maritime Pickup and Delivery Problem with Time Windows and Split Loads. *INFOR: Information Systems and Operational Research*. 2011. Vol. 49, No. 2. pp. 79–91.

[193] M. Cherkesly, G. Desaulniers, G. Laporte. A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading. *Computers & Operations Research*. 2015. Vol. 62. pp. 2015.

[194] S. Naccache, J.-F. Côté, L. C. Coelho. The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*. Mar. 2018.

[195] A. Lim, F. Wang, Z. Xu. The Capacitated Traveling Salesman Problem with Pickups and Deliveries on a Tree. Springer, Berlin, Heidelberg. 2005. pp. 1061–1070.

[196] S. Anily, J. Bramel. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics*. 1999. Vol. 46, No. 6. pp. 654–670.

[197] J.-F. Côté, M. Gendreau, J.-Y. Potvin. Large neighborhood search for the pickup and delivery traveling salesman problem with multiple stacks. *Networks*. 2012. Vol. 60, No. 1. pp. 19–30.

[198] F. Carrabs, R. Cerulli, J.-F. Cordeau. An Additive Branch-and-Bound Algorithm for the Pickup and Delivery Traveling Salesman Problem with LIFO or FIFO Loading. *INFOR: Information Systems and Operational Research*. 2007. Vol. 45, No. 4. pp. 223–238.

[199] G. Erdoğan, J.-F. Cordeau, G. Laporte. The pickup and delivery traveling salesman problem with first-in-first-out loading. *Computers & Operations Research*. 2009. Vol. 36, No. 6. pp. 1800–1808.

[200] A. Malapert, C. Guéret, N. Jussien. Two-dimensional pickup and delivery routing problem with loading constraints. *First CPAIOR Workshop on Bin Packing and Placement Constraints (BPPC'08)*. 2008.

[201] J. L. M. da Silveira, E. C. Xavier. Pickup and Delivery Problem with Two Dimensional Loading/Unloading Constraints. *Computational Logistics: 5th International Conference, ICCL 2014, Valparaiso, Chile, September 24-26. 2014. Proceedings*. Springer International Publishing. 2014. pp. 31–46.

[202] D. Männel, A. Bortfeldt. A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. *European Journal of Operational Research*. 2016. Vol. 254, No. 3. pp. 840–858.

[203] E. E. Zachariadis, C. D. Tarantilis, C. T. Kiranoudis. The Pallet-Packing Vehicle Routing Problem. *Transportation Science*. 2012. Vol. 46, No. 3. pp. 341–358.

[204] D. Männel, A. Bortfeldt. Solving the Pickup and Delivery Problem with 3D Loading Constraints and Reloading Ban. FEMM Working Papers. 2015.

[205] G. Scheithauer, Y. G. Stoyan, T. Y. Romanova. Mathematical Modeling of Interactions of Primary Geometric 3D Objects. *Cybernetics and Systems Analysis*. 2005. Vol. 41, No. 3. pp. 332–342.

[206] I. V. Grebennik, A. V. Pankratov, A. M. Chugay, A. V. Baranov. Packing n-dimensional parallelepipeds with the feasibility of changing their orthogonal orientation in an n-dimensional parallelepiped. *Cybernetics and Systems Analysis*. 2010. Vol. 46, No. 5. pp. 793–802.

[207] Y. G. Stoyan, N. I. Gil, G. Scheithauer, A. Pankratov, I. Magdalina. Packing of convex polytopes into a parallelepiped. *Optimization*. 2005. Vol. 54, No. 2. pp. 215–235.

[208] H. Pollaris, K. Braekers, A. Caris, G. K. Janssens, S. Limbourg. Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*. 2015. Vol. 37, No. 2. pp. 297–330.

[209] F. Wang, Y. Tao, N. Shi. A Survey on Vehicle Routing Problem with Loading Constraints. *2009 International Joint Conference on Computational Sciences and Optimization*. 2009. pp. 602–606.

[210] Мультимодальные и интермодальные пассажирские перевозки - разными видами транспорта. [Електронний ресурс]. 2017. Режим доступу до ресурсу: http://wikitransport.com/transport/intermodal/.

[211] Г. Левкин. Организация интермодальных перевозок. 2014.

[212] S. T. Choong, M. H. Cole, E. Kutanoglu. Empty container management for intermodal transportation networks. *Transportation Research Part E: Logistics and Transportation Review*. 2002. Vol. 38, No. 6. pp. 423–438.

[213] D. Tsamboulas, S. Kapros. Decision-Making Process in Intermodal Transportation. *Transportation Research Record: Journal of the Transportation Research Board*. 2000. Vol. 1707. pp. 2000.

[214] F. Bruns, S. Knust. Optimized load planning of trains in intermodal transportation. *OR Spectrum*. 2012. Vol. 34, No. 3. pp. 511–533.

[215] Y. Bontekoning, C. Macharis, J. Trip. Is a new applied transportation research field emerging? – A review of intermodal rail–truck freight transport literature. *Transportation Research Part A: Policy and Practice*. 2004. Vol. 38, No. 1. pp. 1–34.

[216] A. Falzarano, S. Ketha, J. Hawker. Development of an intermodal network for freight transportation analysis. Rochester Institute of Technology, Jan. 2007.

[217] F. Bruns, M. Goerigk, S. Knust, A. Schöbel. Robust load planning of trains in intermodal transportation. *OR Spectrum*. 2014. Vol. 36, No. 3. pp. 631–668.

[218] N. Boysen, D. Briskorn, S. Knust. Rail terminal operations. *OR Spectrum*. 2018. Vol. 40, No. 2. pp. 317–318.

[219] P. Guo, W. Cheng, Y. Wang, N. Boysen. Gantry crane scheduling in intermodal rail-road container terminals. International Journal of Production Research, pp. 1–18, Mar. 2018.

[220] E. Pesch, N. Boysen, M. Fliedner, F. Jaehn. A Survey on Container Processing in Railway Yards. *Transportation Science*. 2013. Vol. 47, No. 3. pp. 312–329.

[221] E. Pesch, N. Boysen, F. Jaehn. Scheduling Freight Trains in Rail-Rail Transshipment Yards. *Transportation Science*. 2011. Vol. 45, No. 2. pp. 199–211.

[222] N. Boysen, F. Jaehn, E. Pesch. New bounds and algorithms for the transshipment yard scheduling problem. *Journal of Scheduling*. 2012. Vol. 15, No. 4. pp. 499–511.

[223] M. Barketau, H. Kopfer, E. Pesch. A Lagrangian lower bound for the container transshipment problem at a railway hub for a fast branch-and-bound algorithm. *Journal of the Operational Research Society*. 2013. Vol. 64, No. 11. pp. 1614–1621.

[224] Random — Generate pseudo-random numbers — Python 2.7.14 documentation. [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://docs.python.org/2/library/random.html#random.sample.

[225] J. A. Hartigan, M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. Applied Statistics. Vol. 28. No. 1, p. 100. 1979.

[226] A. C. Fabregas, B. D. Gerardo, B. T. Tanguilig III. Enhanced Initial Centroids for K-means Algorithm. *International Journal of Information Technology and Computer Science(IJITCS)*. 2017. Vol. 9, No. 1. pp. 26–33.

[227] A. Chadha, S. Kumar. Extension of K-Modes Algorithm for Generating Clusters Automatically. *International Journal of Information Technology and Computer Science(IJITCS)*. 2016. Vol. 8, No. 3. pp. 51–57.

[228] B. T. Lowerre. The HARPY Speech Recognition System. Carnegie Mellon University. 1976.

[229] 9.7. itertools — Functions creating iterators for efficient looping — Python 2.7.14 documentation. [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://docs.python.org/2/library/itertools.html#itertools.permutations.

[230] M. Mahajan, P. Nimbhorkar, K. Varadarajan. The Planar k-Means Problem is NP-Hard. *Proceedings of the 3rd International Workshop on Algorithms and Computation*. 2009. pp. 274–285.

[231] J. Renaud, F. F. Boctor, J. Ouenniche. A heuristic for the pickup and delivery traveling salesman problem. *Computers & Operations Research*. 2000. Vol. 27, No. 9. pp. 905–916.

ДОДАТОК А

Список публікацій здобувача

1. И. В. Гребенник, А. С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Кибернетика и системный анализ. 2012. № 6 (48). С. 96-105.

2. И.В. Гребенник, О.С. Титова, А.С. Литвиненко. Оптимизация линейной функции на множестве циклических перестановок. Бионика интеллекта. 2012. № 2 (67). С. 8-12.

3. Grebennik I., Lytvynenko O. Random generation of combinatorial sets with special properties. Econtechmod: an international quarterly journal on economics of technology and modelling processes. 2016. vol. 5, no. 4. pp. 43–48.

4. Yuri G. Stoyan, Igor V. Grebennik, Viacheslav V. Kalashnikov, Oleksandr S. Lytvynenko. Enumeration and Generation of Permutations with a Partially Fixed Order of Elements. International Journal of Combinatorial Optimization Problems and Informatics. 2017. Vol. 8, No. 1. pp. 19-30.

5. Rémy Dupas, Igor Grebennik, Oleksandr Lytvynenko, Oleksij Baranov. An Heuristic Approach to Solving the one-to-one Pickup and Delivery Problem with Three-dimensional Loading Constraints. International Journal of Information Technology and Computer Science(IJITCS). 2017. Vol.9, No.10. pp.1-12.

6. Igor Grebennik, Rémy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Scheduling Freight Trains in Rail-rail Transshipment Yards with Train Arrangements. International Journal of Intelligent Systems and Applications (IJISA). 2017. Vol.9, No.10, pp.12-19.

7. И. В. Гребенник, А. С. Литвиненко. Генерация перестановок с частично определенным порядком следования элементов. Бионика интеллекта. 2017. №2 (89). С. 26–30.

8. И.В. Гребенник, А.С. Литвиненко. Генерация комбинаторных множеств с заданными свойствами. Материалы конференции «XIX

International Conference ‘Problems of decision making under uncertainties’». 23-27 апреля 2012. Мукачево, Украина. С. 88-89.

9. I.V. Grebennik, O.S. Lytvynenko, O.S. Titova. Optimization of linear functions on cyclic permutations. Материалы конференции «XX International Conference ‘Problems of decision making under uncertainties’». 17-21 сентября 2012. Брно, Чехия. С. 43-44.

10. I. Grebennik, O. Lytvynenko. Random and exhaustive generation of combinatorial sets with special properties. Материалы конференции «5th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2014)». 24-25 ноября 2014. University of National and World Economy (UNWE). София, Болгария. С. 170-177.

11. R. Dupas, I. Grebennik, O. Lytvynenko. Combinatorial mathematical model and decision strategy for one-to-one pickup and delivery problem with 3D loading constraints. Материалы конференции «X International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT 2015)». 14-17 сентября 2015. Lviv Polytechnic National University. Львов, Украина. С. 133-135.

12. R. Dupas, I. Grebennik, O. Lytvynenko. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «V Konferencja Zastosowań Matematyki w Technice, Informatyce i Ekonomii». 17 сентября 2015. Instytut Matematyki Wydział Matematyki Stosowanej Politechnika Śląska. Гливице, Польша. С. 26-31.

13. O. Lytvynenko, O. Baranov, R. Dupas, I. Grebennik. Three-dimensional one-to-one pickup and delivery routing problem with loading constraints. Материалы конференции «6th International Conference on Information Systems, Logistics and Supply Chain (ILS 2016)». 1-4 июня 2016. Бордо, Франция. С. 101-108.

14. И. В. Гребенник, А.С. Литвиненко. Перечисление и генерация перестановок с частично заданным порядком следования элементов.

Материалы конференции «Сучасні проблеми машинобудування». 21-22 ноября 2016. Харьков, Украина. С. 26.

15. Igor Grebennik, Remy Dupas, Oleksandr Lytvynenko, Inna Urniaieva. Deciding on train arrangement in freight trains scheduling problem. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 35-41.

16. Igor Grebennik, Oleksandr Lytvynenko. Developing software for solving some combinatorial generation and optimization problems. Материалы конференции «7th International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE-2017)». 3-4 ноября 2017. University of National and World Economy (UNWE). София, Болгария. С. 58-64.

ДОДАТОК Б

Акти впровадження результатів дисертаційної роботи

ЗАТВЕРДЖУЮ
Директор
ТОВ "Клауд Воркс"
_____ Д.С. Свербілов
«25» жовтня 2018р.

АКТ
про використання результатів дисертаційної роботи
на здобуття наукового ступеня кандидата технічних наук
Литвиненка Олександра Сергійовича

Комісія у складі голови комісії директора Свербілова Д.С., членів комісії програміста Луцика В.А. та програміста Горносталь О.А. склала даний акт про використання результатів дисертаційної роботи, виконаної Литвиненком Олександром Сергійовичем.

Методи, алгоритми та програмне забезпечення, запропоновані Литвиненком О.С., використовуються в ТОВ «Клауд Воркс» в програмному забезпеченні для розв'язання задач, пов'язаних з перевезенням та обробкою вантажів, зокрема задачі вивозу та доставки вантажів (Pickup and Delivery Problem) та задачі складання розкладу руху вантажних поїздів на сортувальній станції (Transshipment Yards Scheduling Problem).

Програмне забезпечення, розроблене Литвиненком О.С., має дружній інтерфейс та є досить гнучким, що дозволяє використовувати його для рішення зазначених класів задач як у ситуаціях, коли найбільш важливою вимогою є швидкість отримання розв'язку, так і у ситуаціях, коли першим пріоритетом є його точність.

Голова комісії: Свербілов Д.С. _____

Члени комісії: Луцик В.А. _____

Горносталь О.А. _____

УКРАЇНА ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ "КЛАУД ВОРКС" №37462274 МІСТО ХАРКІВ

«ЗАТВЕРДЖУЮ»
Проректор з науково-
методичної роботи
Харківського національного
університету радіоелектроніки

_____ І.В. Рубан

«17» 04 2018 р.

АКТ

про використання результатів дисертаційної роботи Литвиненка Олександра Сергійовича «Методи генерації комбінаторних конфігурацій та їх застосування в математичному і комп'ютерному моделюванні задач перевезення та обробки вантажів» в НДР «Розробка методології і математичних моделей соціально-економічних систем при реалізації концепції їх стійкого розвитку» (ДР № 0115U001522), що виконувалася відповідно до плану науково-дослідних робіт Харківського національного університету радіоелектроніки

Комісія у складі:

Голови: завідувача кафедри системотехніки, наукового керівника ДР № 0115U001522 д.т.н., проф. Гребенніка І.В.,

Членів комісії: заступника начальника наукового інформаційно-аналітичного відділу Хоменко Н.Ю., доцента кафедри системотехніки Губаренка Є.В. встановила, що результати наукових досліджень **Литвиненка Олександра Сергійовича** реалізовано у межах виконання держбюджетної теми ДР № 0115U001522 Харківського національного університету радіоелектроніки.

В рамках робіт, що виконувались відповідно до плану НДР, автором проведено дослідження методів комбінаторної генерації та оптимізації, розроблено базовий метод генерації комбінаторних конфігурацій, розроблено методи повної та випадкової часткової генерації композиційних *k*-образів комбінаторних множин (*k*-множин), запропоновано нову комбінаторну множину – перестановки з частково заданим порядком слідування елементів, розроблено математичні моделі задачі транспортної маршрутизації (Pickup and Delivery Problem з тримірними обмеженнями завантаження) та задачі складання розкладу руху вантажних поїздів на сортувальній станції, що використовують апарат комбінаторних конфігурацій замість традиційних булевих змінних, а також методи їх рішення.

Розроблені комбінаторні оптимізаційні методи дозволяють значно підвищити ефективність рішення зазначених класів задач за рахунок врахування додаткових факторів (відповідно обмежень завантаження та розміщення поїздів на коліях).

Завідувач кафедри системотехніки, проф. д.т.н. І.В. Гребеннік

Заст. начальника наукового
інформаційно-аналітичного відділу Н.Ю. Хоменко

доц. каф. системотехніки, к.т.н. Є.В. Губаренко

ДОДАТОК В. Скріншоти розроблених програмних модулів

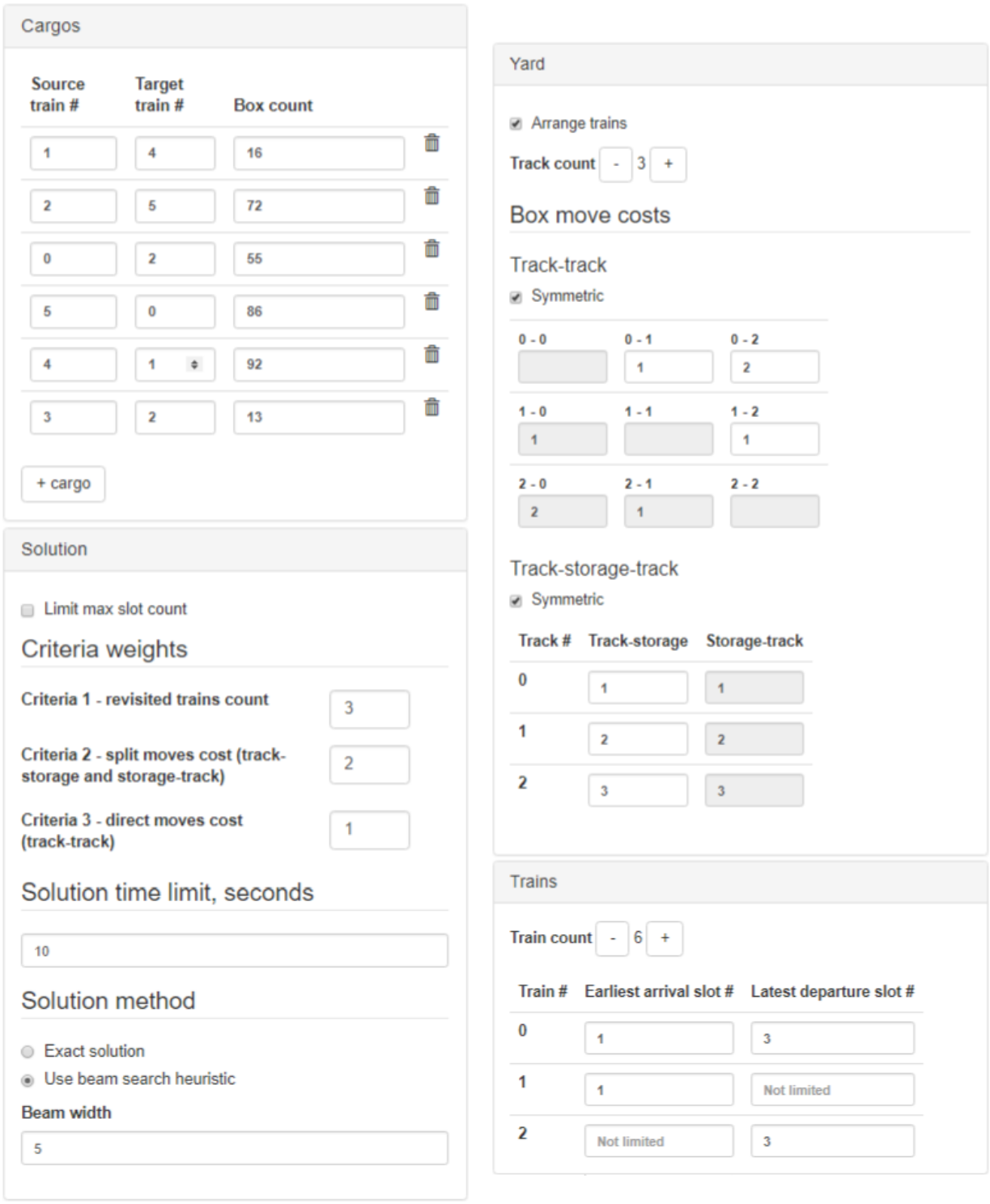


Рисунок В.1. Скріншот розробленого програмного модуля для розв'язання задачі TYSP2 (вхідні дані)

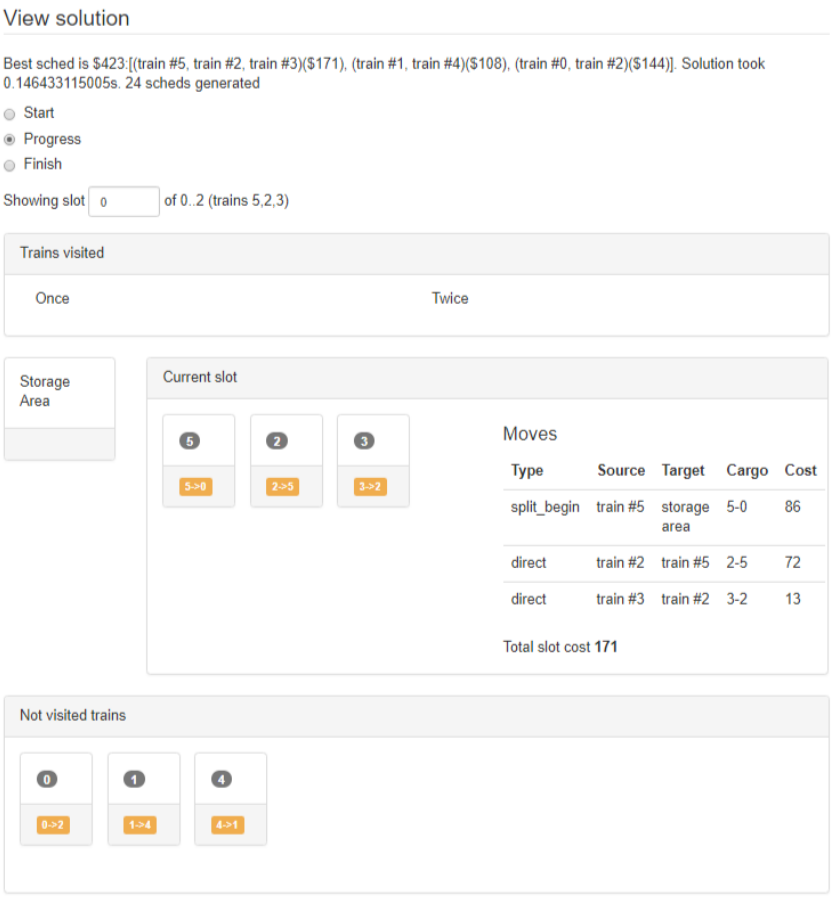


Рисунок В.2. Скріншот розробленого програмного модуля для розв'язання задачі TYSP2 (розв’язання, перший слот)

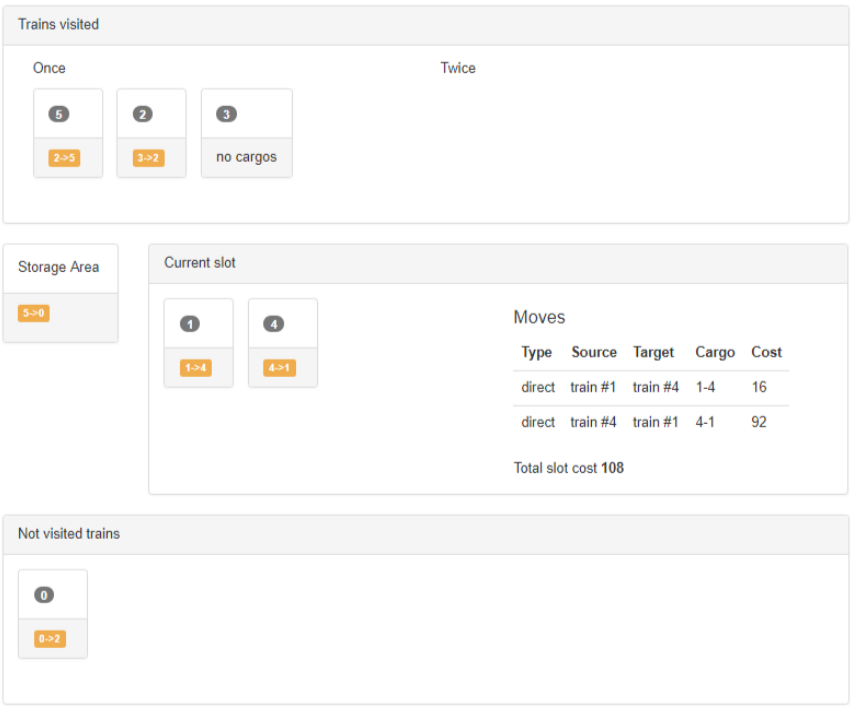


Рисунок В.3. Скріншот розробленого програмного модуля для розв'язання задачі TYSP2 (розв’язання, другий слот)

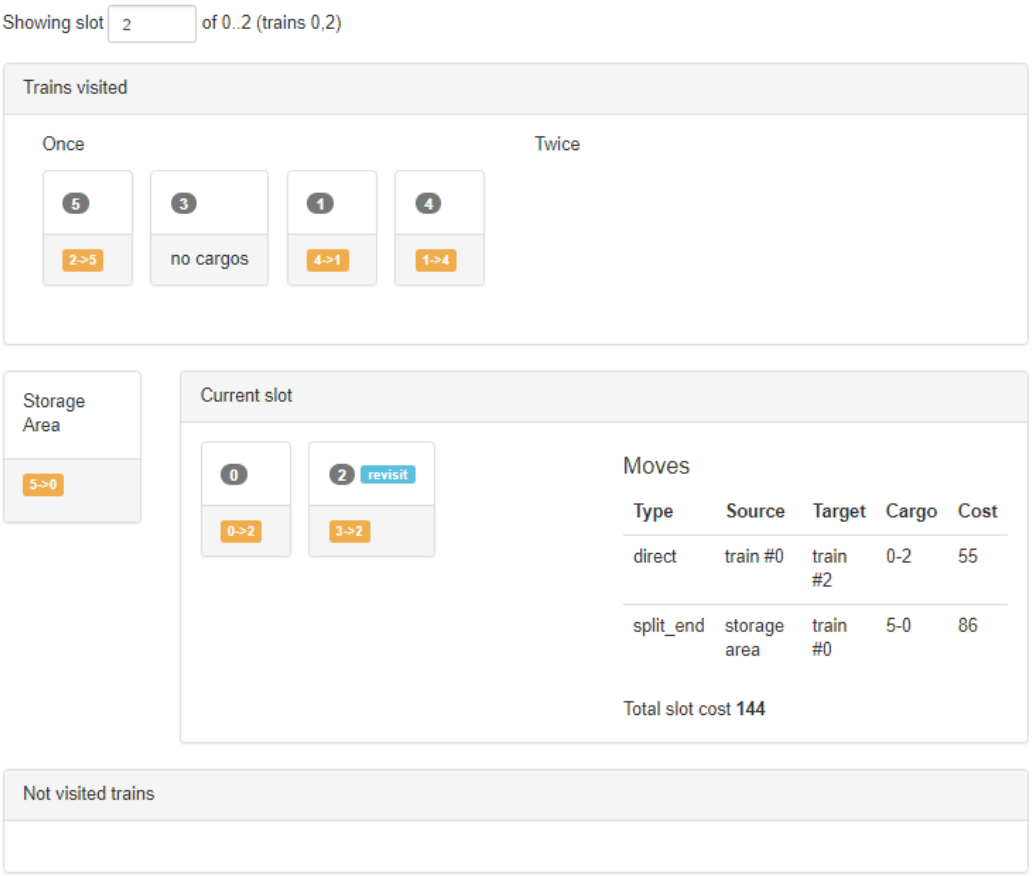


Рисунок В.4. Скріншот розробленого програмного модуля для розв'язання задачі TYSP2 (розв’язання, третій слот)