КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.85 DOI:10.34229/2707-451X.22.4.5

Г.Д. БІЛА, О.О. КОРЧИНСЬКИЙ, П.І. СТЕЦЮК, О.М. ХОМ'ЯК, С.Б. ШЕХОВЦОВ

ВИКОРИСТАННЯ NEOS-СЕРВЕРА ДЛЯ РОЗВ'ЯЗАННЯ ДВОХ КЛАСІВ ОПТИМІЗАЦІЙНИХ ЗАДАЧ

Робота присвячена дослідженню можливостей розв'язання задач оптимізації за допомогою NEOS сервера на прикладі двох задач математичного програмування: задачі лінійного булевого програмування для відомої задачі т-комівояжерів та задачі нелінійного програмування для спеціальної багато екстремальної задачі на многовиді Штіфеля. Для обчислень були використані доступні на NEOS сервері солвери CPLEX, Gurobi та BARON. Проведено порівняння ефективності двох моделей булевого лінійного програмування у залежності від зміни обмежень, пов'язаних зі зв'язністю циклу та порядком обходу його вершин для тестових прикладів з відомої бібліотеки TSPLIB. Для задач на многовиді Штіфеля досліджено постановки залежно від обмежень, накладених на компоненти векторів.

Ключові слова: *оптимізація, NEOS сервер, AMPL, NEOS солвери, задача комівояжера, многовид Штіфеля,*

Вступ. Задачі математичного програмування та чисельні методи їх розв'язання постійно вдосконалюються та змінюються. Розроблені ефективні чисельні методи можуть бути використані для конкретного застосування або для вирішення широкого кола задач оптимізації. Таку можливість надає NEOS (Network-Enabled Optimization System) сервер, який дозволяє користувачу вирішувати оптимізаційні задачі в онлайн режимі. NEOS сервер зв'язує користувача та необхідне сучасне програмне забезпечення для розв'язання оптимізаційних задач. Він приймає описану оптимізаційну задачу та повертає розв'язок в інтерактивному режимі та електронною поштою. Оптимізаційні додатки дозволяють користувачу зосередитися на описі моделі оптимізаційної задачі. Користувачам не потрібно купувати та встановлювати програмне забезпечення для оптимізації.

Мета роботи – дослідження можливостей розв'язання задач оптимізації за допомогою солверів CPLEX, Gurobi та BARON з NEOS сервера на прикладі двох задач математичного програмування: задачі лінійного булевого програмування для відомої задачі *m*-комівояжерів та задачі нелінійного програмування для спеціальної багатоекстремальної задачі на многовиді Штіфеля. Для задачі *m*-комівояжерів розроблені AMPL-програми для розв'язання відповідних задач булевого лінійного програмування з обмеженнями, які дозволяють уникнути підциклів (Miller–Tucker–Zemlin). Проведено порівняння ефективності солверів CPLEX та Gurobi для тестових прикладів з відомої бібліотеки TSPLIB.

Для задач на многовиді Штіфеля досліджено їх формулювання у залежності від обмежень, накладених на компоненти векторів. Для тестових прикладів наведені оптимальні розв'язки та доведено їх єдиність. Проведено дослідження ефективності солвера BARON для знаходження розв'язків залежно від кількості векторів та їх розмірності.

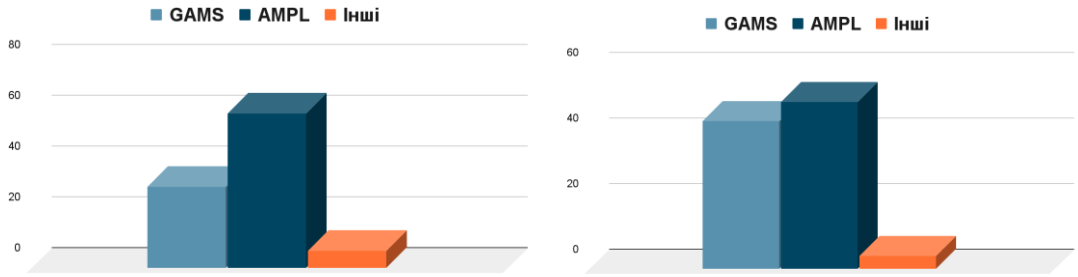
Матеріал роботи викладений у 4 розділах. У розділі 1 описано загальні відомості про використання NEOS сервера для вирішення задач оптимізації та приділено основну увагу мові математичного програмування AMPL (A Mathematical Programming Language) як його складовій частині. Крім того, проведено порівняння, як часто використовується AMPL та мова алгебраїчного моделювання GAMS (General Algebraic Modeling System) порівняно з іншими форматами вхідних даних на сервері. У розділі 2 описано принцип розв'язання задач оптимізації за допомогою NEOS та наведено список наявних на сервері солверів, що підтримують AMPL. Також наведено статистику частоти використання солверів CPLEX, Gurobi та BARON, порівняно з іншими доступними на сервері солверами, та загальну статистику кількості вирішених задач на сервері за 2021 рік. У розділі 3 проведено дослідження задач лінійного булевого програмування для відомої задачі m-комівояжерів та проведено порівняння ефективності розв'язання цих задач за допомогою солверів CPLEX та Gurobi. У розділі 4 проведено дослідження задачі нелінійного програмування для спеціальної багатоекстремальної задачі на многовиді Штіфеля. Для задачі наведено її загальне формулювання та формулювання, залежно від обмежень, накладених на компоненти векторів. Використовуючи солвер BARON, досліджено ефективність розв'язання ним тестових прикладів для векторів з випадковими величинами компонент.

1. Про NEOS сервер та AMPL. NEOS Server [1] – це клієнт-серверна програма в Інтернеті, яка надає безкоштовний доступ до бібліотеки оптимізаційних солверів (програм для розв'язання задач математичного програмування). Задачі розв'язуються на кластері високопродуктивних машин, керованих програмним забезпеченням HTCondor – спеціалізованою системою управління робочим навантаженням для роботи з інтенсивними (високопродуктивними) обчисленнями. Як і інші повнофункціональні пакетні системи, HTCondor надає механізм управління чергою завдань та пріоритетністю їх виконання, моніторинг та управління ресурсами.

Більшість солверів розміщені в Університеті Вісконсина в Медісоні (США). Решта солверів розміщені в партнерських організаціях: Університеті штату Аризона (США), Університеті Клагенфурта (Австрія) та Університеті Мінью (Португалія). Сервер був розроблений у 1996 році Центром технологій оптимізації Аргонської національної лабораторії та Північно-Західного університету (США). Проект NEOS був запущений для розробки сервера з обміну ресурсами програмного забезпечення для оптимізації через мережу Інтернет. NEOS сервер став одним із перших прикладів надання послуг з використання програмного забезпечення. Детальну інформацію про архітектуру і можливості початкової версії сервера можна знайти у [2–4].

NEOS сервер обробляє моделі задач оптимізації, описані мовами моделювання, мовами програмування та відомими форматами даних для задач математичного програмування. Детально про доступні формати даних на NEOS описано в [1]. Більшість солверів лінійного, цілочислового та нелінійного програмування працюють з вхідними даними AMPL та/або GAMS, статистика використання яких протягом року (з 01.07.2021 до 30.06.2022), показана на рис. 1.

На рис. 1,*а* показано відношення кількості надісланих робіт мовами моделювання AMPL та GAMS до загальної їх кількості, що становить 1 376 405. Серед них 834 643 роботи використали AMPL, що складає 61% від загальної кількості. GAMS використовувався для 444 365 робіт – 32%. Іншими форматами вхідних даних скористалися у 7% робіт. На рис. 1,*б* показано статистику використання AMPL та GAMS на сервері за 4 квартал 2021 року, протягом якого різниця між використанням AMPL та GAMS була мінімальною. Протягом 4 кварталу сервером було оброблено 290 402 задач мовою AMPL та 254 500 задач мовою GAMS.



а *б*

РИС. 1. Статистика використання AMPL, GAMS та інших форматів вхідних даних

Мова математичного програмування AMPL – це мова алгебраїчного моделювання для опису математичних моделей оптимізаційних задач та алгоритмів їх розв'язання. AMPL розробили Роберт Фоурер, Девід Гай та Браян Керніган (Bell Laboratories) у 80-х роках [5]. Синтаксис AMPL схожий на звичайний опис математичних моделей оптимізації, в якому описуються дані задачі, невідомі змінні, цільові функції та обмеження оптимізаційної задачі.

Для розв'язання задач на сторінці кожного солвера є вебформа, де кожен файл ("модель", "вхідні дані", "команди") має відповідати вибраному формату опису задачі. Користувач має можливість заповнити у вебформі модель задачі, вхідні дані та файл з командами, що мають виконуватись для описаної моделі задачі оптимізації (рис. 2). Інформація про модель задачі оптимізації, її вхідні дані та команди управління моделлю та солвером можуть подаватися як в окремих файлах, так і об'єднано у файлі моделі. Вебформа надає можливість додати коментар, що відображатиметься єдиним блоком при отриманні результату обчислень. У додаткових налаштуваннях є можливість отримати згенеровану роботу у форматі XML без надсилання задачі на її розв'язання на NEOS, що широко використовується для структурування та передачі даних. Також доступна опція, за допомогою якої можна підвищити пріоритет виконання роботи на NEOS за умови, що максимальний час виконання не перевищує 5 хвилин.

Основний механізм вирішення задач оптимізації за допомогою NEOS сервера полягає у тому, що користувач пише специфікацію для певної проблеми та надсилає її через один із встановлених інтерфейсів. При отриманні сервер надає завданню його номер, пароль і ставить у чергу для проведення обчислень на одному з віддалених комп'ютерів. Як тільки завдання отримане, воно розпаковується, драйвер обробляє його, включаючи розрахунки. Обробка завдання включає обчислення необхідних похідних та зв'язування бібліотек алгоритмів та об'єктів; далі виконується код оптимізаційного солвера, на що відводиться максимум 8 годин. Після закінчення обчислень віддалений комп'ютер повідомляє результати на сервер, який передає їх у інтерфейс користувача. У залежності від солвера інтерфейс користувача відображає оптимальний розв'язок, який супроводжується такою технічною інформацією як: кількість використаних змінних та обмежень, нижня і верхня межі оптимального розв'язку та інші дані, передбачені конкретним солвером.

Завдання на NEOS сервер можна надсилати через веб-сторінку, електронну пошту, XML RPC, Kestrel або опосередковано через інструменти сторонніх розробників: SolverStudio для Excel, OpenSolver, Pyomo, JuMP і R-пакет rneos. Серверна частина реалізована на Python3 і використовує XML-RPC як базовий протокол зв'язку для обробки запитів і відповідей HTTP.

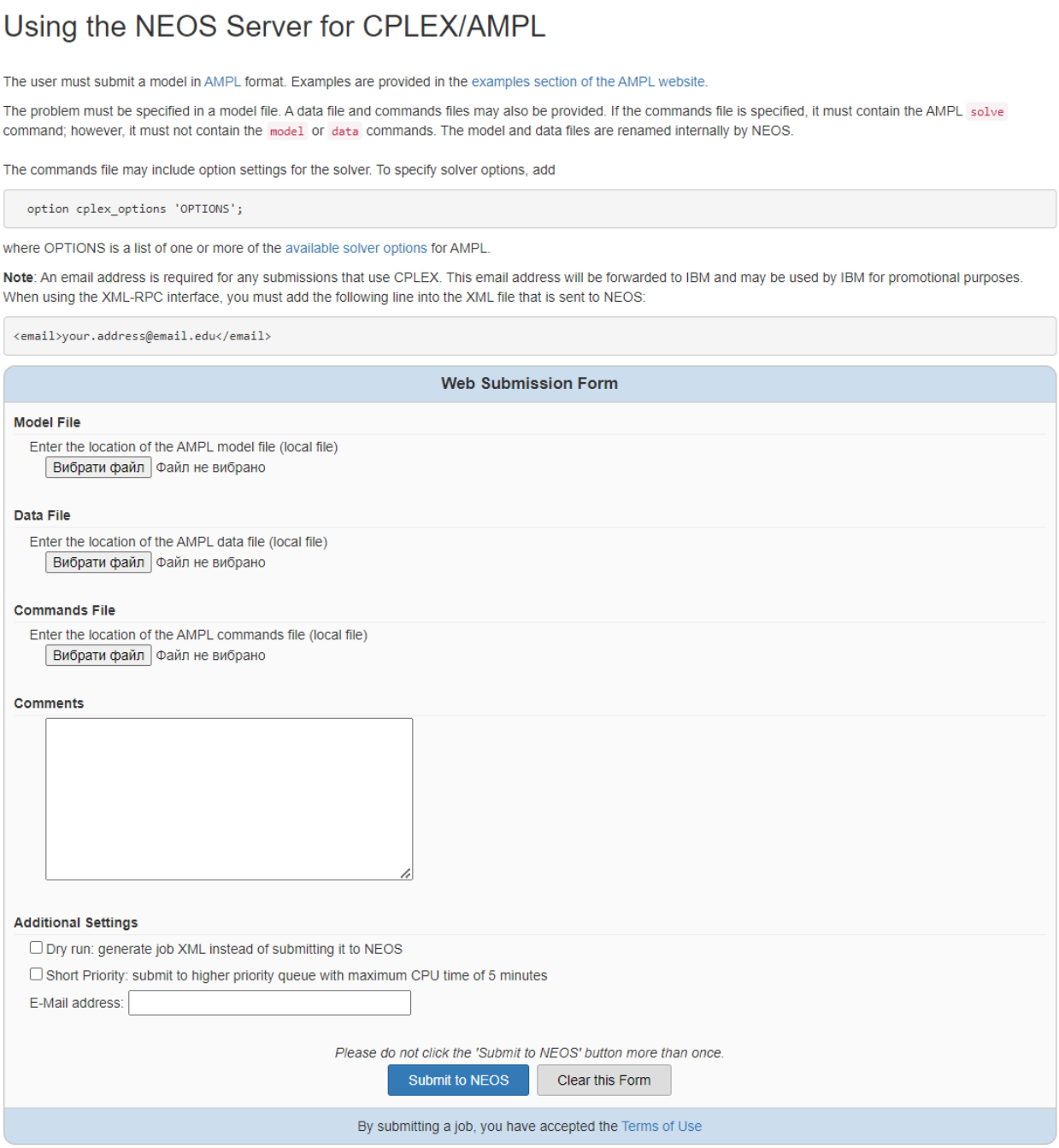


РИС. 2. Веб-форма для надсилання завдань на NEOS за допомогою солвера CPLEX та мови AMPL

2. Про солвери NEOS сервера. NEOS надає доступ до більш ніж 90 солверів у 17 видах задач оптимізації (класи оптимізаційних задач). Користувачі сервера вибирають солвер зі списку доступних та надсилають оптимізаційну задачу через один із наявних інтерфейсів. NEOS сервер забезпечує розв'язання оптимізаційної задачі за допомогою вибраного солвера та повертає розв'язок як

описано у попередньому розділі. Для кожного солвера NEOS вказані ті формати вхідних даних, які можуть використовуватися для розв'язання оптимізаційної задачі. Приклад використання AMPL для солвера CPLEX [6] показано на рис. 2.

Для використання вибраного солвера та відповідного формату вхідних даних NEOS сервер надає окрему сторінку, яка для солвера Gurobi [7] має такий вигляд (рис. 3).

РИС. 3. Сторінка солвера Gurobi на NEOS сервері

На сторінці кожного солвера надається інформація про види задач оптимізації та формати вхідних даних, які підтримує вибраний солвер. На сторінці також наявна вебформа для подання задачі на розв'язання, що підтримує визначений користувачем формат вхідних даних.

Далі наведено солвери з NEOS сервера, які використовують AMPL для відповідних видів задач математичного програмування:

1. Bound Constrained Optimization: **L-BFGS-B**;
2. Complementarity Problems**: Knitro, PATH**;
3. Global Optimization: **ASA, BARON, Couenne, icos, LGO, OCTERACT, PGAPack, PSwarm, RAPOSa, scip**;
4. Linear Programming: **bpmpd, COPT, CPLEX, FICO-Xpress, Gurobi, MOSEK, OCTERACT, OOQP**;
5. Mathematical Programs with Equilibrium Constraints: **filterMPEC, Knitro**;
6. Mixed Integer Linear Programming: **Cbc, COPT, CPLEX, feaspump, FICO-Xpress, Gurobi, MINTO, MOSEK, OCTERACT, qsopt_ex, scip;**
7. Mixed Integer Nonlinearly Constrained Optimization: **AlphaECP , ANTIGONE, BARON, Bonmin, Couenne, DICOPT, FilMINT, Knitro, LINDOGlobal, MINLP, OCTERACT, SBB, scip, SHOT**;
8. Mixed-Integer Optimal Control Problems**: MUSCOD-II**;
9. Nondifferentiable Optimization**: condor**;
10. Nonlinearly Constrained Optimization: **CONOPT, filter, Ipopt, Knitro, LANCELOT, LOQO, MINOS, OCTERACT, SNOPT**;
11. Second Order Conic Programming**: CPLEX, FICO-Xpress, Gurobi, scip**;
12. Semi-infinite Optimization**: nsips**.

На рис. 4 показана поквартальна частота використання солверів CPLEX, Gurobi, BARON [8–10] та інших протягом 3 – 4 кварталів 2021 року та 1 – 2 кварталів 2022 року.

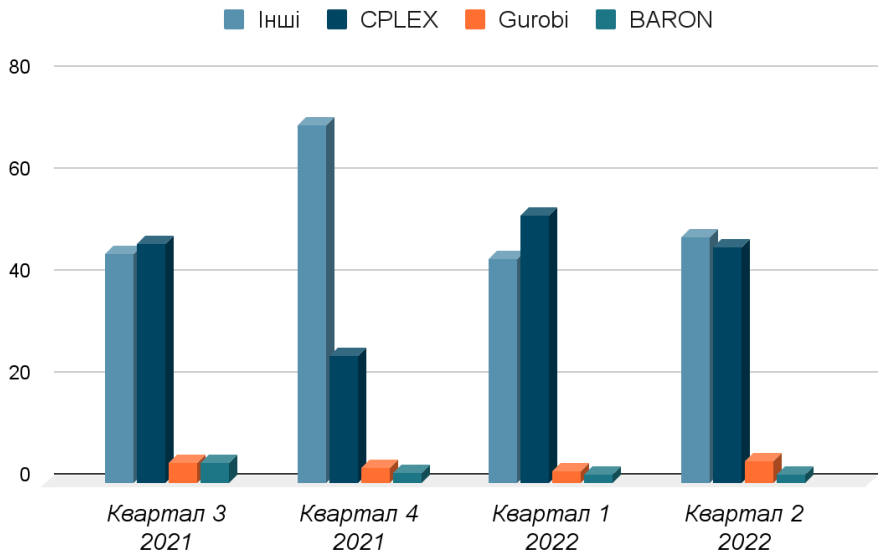


РИС. 4. Використання солверів CPLEX, Gurobi та BARON

За вказаний період CPLEX активно використовували протягом 3 кварталу 2021 року та 1–2 кварталів 2022, менш активно – 4 квартал 2021 року, відсоток використання якого знаходився у проміжку 25 – 52 %. Gurobi та BARON займали у середньому 2 – 4 % залежно від кварталу.

На рис. 5 показано статистику надсилання оптимізаційних задач на NEOS сервер за період з 01.07.2021 до 30.06.2022.

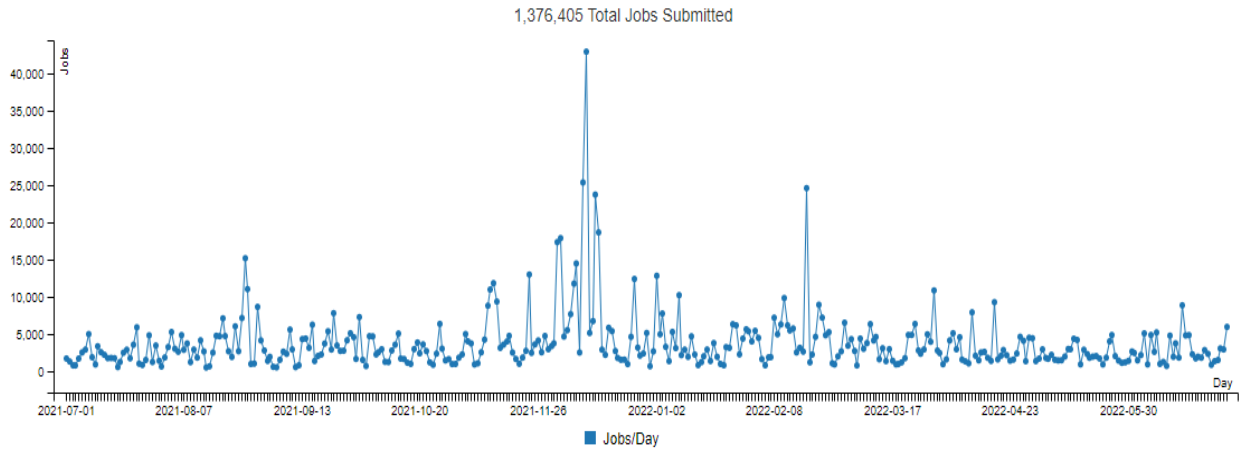


РИС. 5. Статистика розв'язаних задач з 01.07.2021 до 30.06.2022

У центрі рисунку відображений період осінь–зима–весна і видно, що пік завантаження NEOS сервера приходиться на грудень, що може бути пов'язано з активністю студентів та аспірантів під

час закінчення першого семестру. За вказаний період NEOS сервер у середньому обробляв 3771 робіт у день, максимальне завантаження прийшлося 11 грудня 2021 року – 42 953 роботи.

Далі розглянемо використання солверів CPLEX та Gurobi для розв'язання задачі лінійного булевого програмування для відомої задачі *m*-комівояжерів та використання солвера BARON для розв'язання багатоекстремальної задачі нелінійного програмування для спеціальної задачі на многовиді Штіфеля.

3. CPLEX та Gurobi: задача *m*-комівояжерів. Ця задача є узагальненням відомої задачі комівояжера, де дозволяється використовувати більше одного комівояжера. Нехай маємо *m* комівояжерів. У випадку, якщо $m=1$, то цю задачу можна розглядати як класичну задачу комівояжера. Зауважимо, що на NEOS сервері представлений окремий солвер concorde, який спеціалізується на розв'язанні класичної задачі комівояжера.

Постановка задачі. Дано n міст. Позначимо їх $x_1, x_2, \ldots, x_n$. Відстані c_{ij} між будь-якою парою міст x_i та x_j відомі, $1 \le i, j \le n$. У одному з міст знаходяться m комівояжерів, кожен з яких виїжджає з початкового міста, об'їжджаючи кожне місто тільки один раз та повертаючись у вихідне місто. Мета комівояжерів полягає у визначенні такого набору m маршрутів, що не перетинаються, щоб мінімізувати загальну вартість маршруту.

У задачі m-комівояжерів потрібно врахувати такі вимоги до набору m маршрутів:

- усі маршрути мають починатися і закінчуватися в одному і тому ж місті;
- кожен комівояжер може відвідати декілька міст;
- кожне місто має бути відвіданим лише один раз.

Математична модель. Введемо n^2 булевих змінних x_{ij}, $i=\overline{1,n}$, $j=\overline{1,n}$, для яких $x_{ij}=1$, якщо комівояжер їде з міста i у місто j, інакше $x_{ij}=0$. Тоді сумарна вартість маршруту комівояжера становить $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$. Виїзд m комівояжерів з початкового міста і повернення у нього задається умовами:

$$\sum_{j=1}^{n} x_{1j} = m, \quad \sum_{i=1}^{n} x_{i1} = m.$$

Одноразовий виїзд та в'їзд комівояжерів із усіх міст окрім початкового задається умовою:

$$\sum_{j=1}^{n} x_{ij} = 1,\ i=\overline{2,n}, \quad \sum_{i=1}^{n} x_{ij} = 1,\ j=\overline{2,n}.$$

Проте цих умов недостатньо – вони не забороняють підцикли. Щоб позбутися підциклів, використаємо міркування [11–13]: введемо *n* цілих змінних u_i, які будуть відповідати порядковим номерам міст, які відвідає комівояжер, та параметр p, який буде вказувати максимальну кількість міст, яку може відвідати один комівояжер. Для заборони підциклів використовуються обмеження

$$u_i - u_j + px_{ij} \le p-1,\ i, j=\overline{2,n},\ i \ne j,$$

які для задачі комівояжера вперше були запропоновані у роботі [11] С. Міллером, А. Таккером і Р. Земліним (Miller – Tucker – Zemlin).

В результаті отримаємо наступну математичну модель задачі *m* комівояжерів:

$$d_p = \min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

при таких обмеженнях:

$$\sum_{j=1}^{n} x_{1j} = m, \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{2,n}, \tag{2}$$

$$\sum_{i=1}^{n} x_{i1} = m, \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{2,n}, \tag{3}$$

$$u_i - u_j + p x_{ij} \le p - 1, \; i, j = \overline{2,n}, \; i \ne j, \tag{4}$$

$$x_{ij} \in \{0,1\}, \; 1 \le u_i \le n-1, \quad i = \overline{2,n}, \; u_i \; - \text{цілі числа}, \tag{5}$$

де (1) – цільова функція; умовам виїзду з кожного міста відповідають обмеження (2); умовам в'їзду в кожне з міст відповідають обмеження (3); за зв'язність циклу та порядок відвідуваності його вершин відповідають обмеження (4); за булеві змінні та порядкові номери обходу циклу відповідають обмеження (5).

Покажемо, що, якщо $m = 1$, то обмеження (4) гарантують наявність рівно одного циклу, що проходить через усі міста. Нехай є 2 цикли. Один із них не проходить через перше місто, позначимо його $(i_2, \ldots, i_p, i_2)$. Випишемо для кожної пари міст, що йдуть по порядку в цьому циклі, умови (4):

$$\begin{gathered} u_{i2} - u_{i3} + p \le p - 1 \\ u_{i3} - u_{i4} + p \le p - 1 \\ \vdots \\ u_{ip} - u_{i2} + p \le p - 1. \end{gathered}$$

Додавши ці нерівності, отримаємо $pn \le p(n-1)$, що неможливо при $p \ne 0$.

Перевіримо, що якщо є цикл, що проходить через усі міста, то він задовольняє умові (4), тобто можна підібрати відповідні значення змінних u_i. Нехай $u_i = p$, якщо місто i відвідали на кроці p, тоді нерівність $u_i - u_j \le n-1$ правильна при $x_{ij} = 0$. Якщо $x_{ij} = 1$, то $u_i = p$, а $u_j = p+1$, тоді $p - (p+1) + n \le n-1$, $n-1 \le n-1$, тобто обмеження (4) виконується як строга рівність. Враховуючи обмеження (2), (3) отримуємо зв'язність циклу.

AMPL-код для опису задачі *m*-комівояжерів має наступний вигляд:

```
# Опис параметрів задачі m-комівояжерів
param n integer >= 3; #кількість міст
param m integer >= 1; #кількість комівояжерів
param p integer >= 1, <= n; #максимум міст для одного комівояжера
param coords{i in 1..n, j in {1, 2}}; #координати міст
param c{i in 1..n, j in 1..n} = round(sqrt((coords[i,1] - coords[j,1])^2 +
(coords[i,2] - coords[j,2])^2)); #матриця відстаней між містами

# Опис змінних задачі m-комівояжерів
var x {i in 1..n, j in 1..n} binary; # матриця переміщень
var u {1..n} >= 1, <= n - 1 integer; #нумерація міст при обході
```

```
# Опис задачі (1)-(5)
minimize obj1: sum {i in 1..n, j in 1..n} c[i,j]*x[i,j]; #цільова функція
# m-комівояжерів повертаються в перше місто
subject to con2a{j in 1..n: j == 1}: sum{i in 1..n} x[i,j]=m;
# в усі міста, крім першого, заїжджає тільки один комівояжер
subject to con2b{j in 1..n: j >= 2}: sum{i in 1..n: i != j} x[i,j]=1;
# з першого міста виїжджає m-комівояжерів
subject to con3a{i in 1..n: i == 1}: sum{j in 1..n} x[i,j]=m;
# з усіх міст, крім першого, виїжджає тільки один комівояжер
subject to con3b {i in 1..n: i >= 2}: sum{j in 1..n: i != j} x[i,j]=1;
# кожен комівояжер відвідує максимум p міст
subject to con4{i in 1..n, j in 1..n: i !=j and i>=2 and j>=2}: u[i] - u[j] +
p*x[i,j] <= p - 1;
```

Для цільової функції та обмежень в AMPL-коді використано наступні назви: `obj1` – цільова функція (1); `con2a` – перша частина обмежень (2); `con2b` – друга частина обмежень (2);. `con3a` – перша частина обмежень (3); `con3b` – друга частина обмежень (3). `con4` – обмеження (4).

Розглянемо застосування вищенаведеного AMPL-коду для тестового прикладу задачі *m*-комівояжерів, де $m = 2$, а вхідні дані, запуск відповідного солвера та відображення результатів розрахунку мають наступний вигляд:

```
data; # вхідні дані
param n := 30; # потрібно відвідати 30 міст
param m := 2; # наявні два комівояжери
param p := 22; # максимальна кількість міст для відвідування комівояжером
param coords : 1 2 := # координати міст: номер_міста, x-координата, y-
координата
 1 111 158     2 6 208     3 75 446     4 41 370     5 340 48
 6 644 1000    7 55 930    8 706 448    9 201 53    10 530 263
11   4  835   12 486 51   13 1000 751  14 717 664  15 77 684
16 510 1000   17 1 323    18 337 97    19 989 991  20 11 49
21 965  277   22 610 718  23 809 995   24 985 871  25 852 33
26 365  987   27 317 984  28 25 988    29 512 8    30 84 914;
display n, m, p;
solve;
display obj1, _solve_elapsed_time;
```

Результат роботи солвера Gurobi на NEOS сервері має такий вигляд:

```
************************************************************
   NEOS Server Version 6.0
   Solver    : lp:Gurobi:AMPL
   Start     : 2022-08-29 06:53:33
   End       : 2022-08-29 07:17:05
   Host      : prod-sub-1.neos-server.org
************************************************************
The license for this AMPL processor will expire in 3.5 days.
Gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-6.neos-server.org
n = 30
m = 2
p = 22
```

```
Presolve eliminates 0 constraints and 30 variables.
Adjusted problem:
900 variables:
      871 binary variables
      29 integer variables
872 constraints, all linear; 4178 nonzeros
      60 equality constraints
      812 inequality constraints
1 linear objective; 870 nonzeros.

Gurobi 9.1.2: threads=4
Gurobi 9.1.2: optimal solution; objective 5489
55608219 simplex iterations
5976888 branch-and-cut nodes
obj1 = 5489
_solve_elapsed_time = 1397.05
```

На рис. 6 показано оптимальні маршрути обох комівояжерів, які починаються та закінчуються у вершині #1.

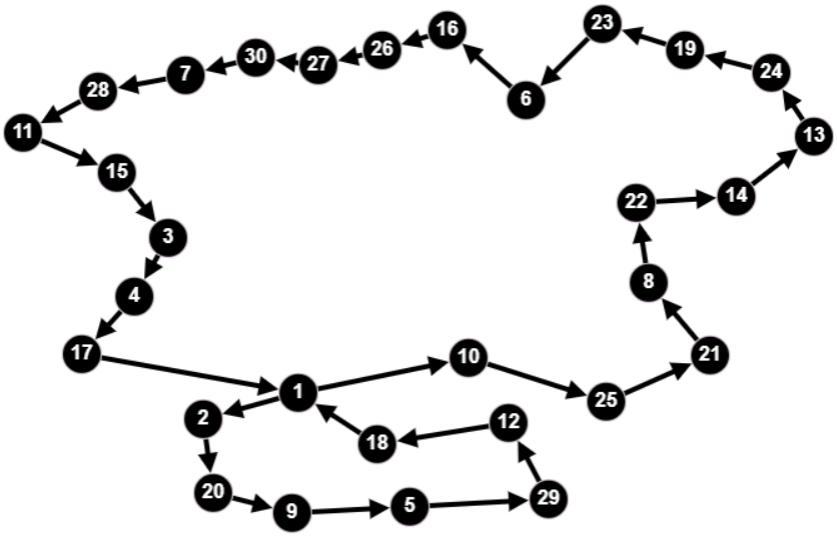


РИС. 6. Розв'язок задачі (1) – (5) для тестового прикладу з двома комівояжерами

Перший оптимальний маршрут є коротким та проходить через 7 вершин, не враховуючи першу вершину. Другий оптимальний маршрут є набагато довшим та включає 22 вершини. Для їх знаходження солверу Gurobi знадобилося 1397.05 секунди.

Перша серія обчислювальних експериментів була присвячена вивченню довжин маршрутів, у залежності від значення параметра *p*, що відображає максимальну кількість міст, яку може відвідати один комівояжер, без урахування початкового міста. Розглядалися варіанти задачі (1) – (5) для двох, трьох та чотирьох комівояжерів на тому ж графі, що і попередня задача. Для розв'язання задачі (1) – (5) використовувався солвер *CPLEX*. Результати експериментів наведені у табл. 1, де t_m – час знаходження розв'язку задачі (колонки 2, 5, 8); d_m – оптимальне значення цільової функції для m комівояжерів (колонки 3, 6, 9); l_{mi} – довжини циклів, утворених i-им комівояжером, $i \le m$ (колонки 4, 7, 10).

Отримані результати експеримента вказують на те, що для даного прикладу з метою зменшення вартості маршрутів вигідно залишати одного комівояжера у початковому місті при $m > 2$. За наявності трьох комівояжерів, задача зводиться до задачі з двома комівояжерами при $p \le 28$, залишаючи одного в початковому місці.

ТАБЛИЦЯ 1. Перша серія експериментів: залежність довжин маршрутів від параметра *p*, солвер CPLEX

p	t_2	d_2	l_{21} / l_{22}	t_3	d_3	l_{31} / l_{32} / l_{33}	t_4	d_4	l_{41} / l_{42} / l_{43} / l_{44}
30	0.2	4834	1 / 30	0.1	4939	1 / 2 / 29	3	5138	1 / 2 / 2 / 28
29	0.2	4834	1 / 30	0.1	4939	1 / 2 / 29	12	5138	1 / 2 / 2 / 28
28	0.5	4939	2 / 29	0.9	4939	1 / 2 / 29	5	5138	1 / 2 / 2 / 28
27	4.6	5033	3 / 28	0.8	5033	1 / 3 / 28	3	5138	1 / 2 / 2 / 28
26	128	5166	4 / 27	0.8	5166	1 / 4 / 27	22	5262	1 / 2 / 3 / 27
25	142	5264	5 / 26	32.8	5264	1 / 5 / 26	22	5369	1 / 2 / 4 / 26
24	736.5	5367	6 / 25	508.3	5367	1 / 6 / 25	80	5463	1 / 3 / 4 / 25
23	1261.5	5395	7 / 24	166	5395	1 / 7 / 24	295	5491	1 / 2 / 6 / 24
22	1397.05	5489	8 / 23	803.1	5489	1 / 8 / 23	831	5585	1 / 3 / 6 / 23
21	10626	5649	9 / 22	4096.2	5649	1 / 9 / 22	1272	5745	1 / 4 / 6 / 22
20	12443	5681	10 / 21	11775	5681	1 / 10 / 21	1123	5786	1 / 2 / 10 / 20
19	14566	5690	11 / 20	13079	5690	1 / 11 / 20	1080	5786	1 / 2 /10 / 20
18	13173	5784	12 / 19	6808.3	5784	1 / 12 / 19	1544	5880	1 / 3 / 10 / 19

Оскільки кожна задача вирішувалась окремо, то некоректно порівнювати час виконання у залежності від кількості комівояжерів, тому що задачі могли вирішуватися на комп'ютерах різної потужності. Для оцінки часу розв'язання кожної окремої задачі було зроблено наступне: задача запускалася на NEOS сервері 5 разів протягом 10 хвилин; часом її виконання вважався усереднений час. Для задачі при $m = 3$, $p = 22$ затрачений час становить 803.1 секунди, при цьому максимальний час був рівний 1144.9 сек., а мінімальний – 535.2 сек.

Друга серія обчислювальних експериментів проводилася для класичної задачі комівояжера, який відвідує $n-1$ місто ($p = n$). Вона є частковим випадком задачі (1) – (5), якщо $m = 1$. У результаті отримаємо таку математичну модель задачі комівояжера [14]:

$$d_1 = \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{6}$$

при обмеженнях:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1,n}, \tag{7}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{1,n}, \tag{8}$$

$$u_i - u_j + n x_{ij} \le n-1, \; i, j = \overline{2,n}, \; i \ne j, \tag{9}$$

$$x_{ij} \in \{0,1\}, \tag{10}$$

$$1 \le u_i \le n-1, \;\; i = \overline{2,n}, \; u_i \; - \text{цілі числа.} \tag{11}$$

Для задачі (6) – (11) для попереднього тестового прикладу на рис. 7 показано оптимальний маршрут одного комівояжера, довжина якого становить 4834 од.

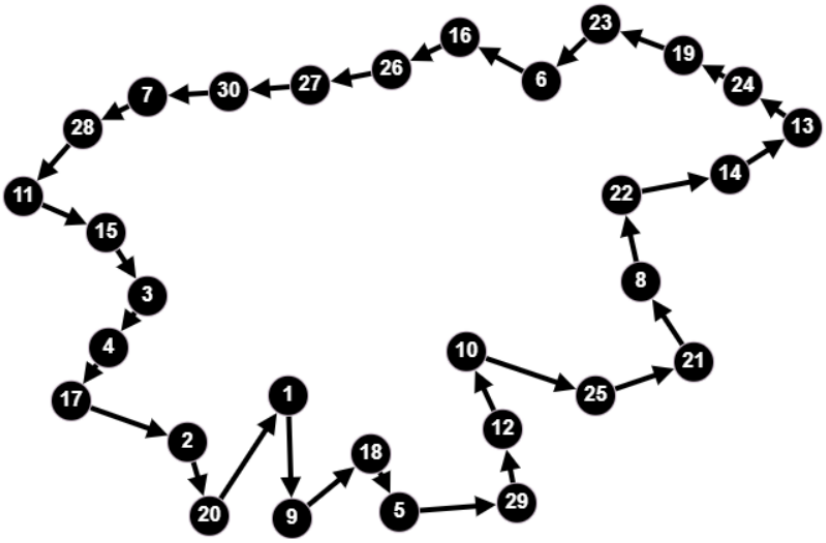


РИС. 7. Оптимальний маршрут комівояжера для тестового прикладу

Зауважимо, що у задачі (6) – (11) обмеження (9) можуть бути замінені на більш тісні обмеження

$$u_i - u_j + (n-1) x_{ij} \le n-2, \; i, j = \overline{2,n}, \; i \ne j, \tag{12}$$

які запропоновані в [15]. Якщо обмеженням (9) відповідає $p = n$, то обмеженням (12) відповідає $p = n-1$.

Друга серія експериментів полягала у порівнянні ефективності солверів *CPLEX* та *Gurobi* для розв'язання задачі (6) – (11) та цієї ж задачі, де обмеження (9) було замінено на обмеження (12). Для проведення експериментів були використані приклади задачі комівояжера з відомої бібліотеки

TSPLIB: st70, eil76, eil101, kroA100 та kroE100 [16]. Порівняння солверів *CPLEX* та *Gurobi* проводилися для задачі (6) – (10), де не вимагалось цілочисельності змінних u_i.

Результати, наведені в табл. 2, де d_1^1 – оптимальне значення цільової функції (6), t_1 – час для розв'язання задачі (6) – (11) або (6) – (10), d_1^2 та t_2 – значення цільової функції (6) та час для розв'язання задачі з використанням обмеження (12).

ТАБЛИЦЯ 2. Друга серія експериментів: порівняння ефективності солверів CPLEX та Gurobi

Задача (6) – (11) та задача (6) – (8), (12), (10), (11)								
Задача	**CPLEX**				**Gurobi**			
	d_1^1	t_1	d_1^2	t_2	d_1^1	t_1	d_1^2	t_2
st70.tsp	675	**3576.5**	675	4499	675	634.5	675	**308.7**
eil76.tsp	538	6880.3	538	**3679.5**	538	3478.6	538	**2070.8**
eil101.tsp	629	92.4	629	**8.2**	629	67.6	629	**35.4**
kroA100.tsp	21282	**877**	21282	3841	21282	**1079**	21282	1277
kroE100.tsp	22068	**6616.8**	22068	7012.8	22068	2536.9	22068	**237.1**
Задача (6) – (10) та задача (6) – (8), (12), (10)								
Задача	**CPLEX**				**Gurobi**			
	d_1^1	t_1	d_1^2	t_2	d_1^1	t_1	d_1^2	t_2
st70.tsp	675	**1231.4**	675	1711.5	675	480	675	**329.2**
eil76.tsp	538	**1816.5**	538	1868.8	538	**4399**	538	7802
eil101.tsp	629	283.1	629	**33.9**	629	47.4	629	**9.1**
kroA100.tsp	21282	**587**	21282	1028	21282	**259**	21282	344
kroE100.tsp	22068	2743.9	22068	**1896.6**	22068	640.6	22068	**571.3**

Для кожного з п'яти тестових прикладів розв'язання відповідних задач (задача (6) – (11), задача (6) – (8), (12), (10), (11), задача (6) – (10), задача (6) – (8), (12), (10)) проводилося на одному й тому ж комп'ютері, на який направляв його NEOS сервер. Це забезпечує коректність порівняння часу їх розв'язання для солверів *CPLEX* та *Gurobi*.

З табл. 2 бачимо, що для кожного з тестових прикладів значення цільових функцій всіх задач співпадають. Задача з використанням обмежень (9) розв'язувалася солвером *CPLEX* швидше, ніж задача з використанням обмежень (12). За допомогою солвера *Gurobi* задачі з використанням обмежень (12) у більшості випадків розв'язувалися швидше, ніж задачі з використанням обмежень (9). Задачі, де не використовувалися обмеження на цілочисельність u_i, переважно розв'язувалися швидше від тих, де ці обмеження враховувались. Варто зауважити, що *Gurobi* у більшості випадків переважав *CPLEX* у швидкості розв'язання відповідних задач.

4. BARON: Квадратична задача на многовиді Штіфеля. Многовид Штіфеля позначається $M_{n,k}$ та складається з усіх ортонормованих систем векторів $\vec{x}_1, \vec{x}_2, ..., \vec{x}_k \in E^n$, де E^n – n-вимірний евклідів простір і $k \le n$. У статті [17] розглядається задача глобальної оптимізації, пов'язана зі знаходженням мінімуму спеціального виду квадратичної однорідної функції на многовиді Штіфеля, яка має наступний вигляд:

$$\min\left\{ f\left(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k\right) = \sum_{i=1}^{k} \vec{x}_i^T A_i \vec{x}_j \right\} \tag{12}$$

при обмеженнях:

$$\vec{x}_i^T \vec{x}_j = \delta_{ij}, \quad 1 \le i, j \le k, \tag{13}$$

$$x_i \in E^n,\ i = \overline{1,k},\ n \ge 2, \tag{14}$$

де A_i, $i = 1,\ldots,k$ – задані дійсні симетричні матриці розміру $n \times n$, δ_{ij} – символ Кронекера. Обмеження (13) – (14) задають многовид Штіфеля $M_{n,k}$.

Задача (12) – (14) – багатоекстремальна задача нелінійного програмування. Її особливості пов'язані зі специфікою цільової функції (12), а саме з тим, що функція $f\left(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k\right)$ не містить членів виду $\vec{x}_i^T A_i \vec{x}_j$ для $i \ne j$. У задачі (12) – (14) потрібно мінімізувати спеціального виду квадратичну функцію при умовах, які задаються $\dfrac{k \cdot (k+1)}{2}$ квадратичними обмеженнями. Розв'язок цієї задачі можна отримати за допомогою пакету BARON як для загального вигляду задачі (12) – (14), так і її окремих випадків, у залежності від того, які умови накладено на компоненти векторів.

У роботі [18] виділяють наступні варіанти задачі (12) – (14):

– (R) компоненти векторів $\vec{x}_i$, $i = 1,\ldots,k$ належать множині дійсних чисел;

– (Z) компоненти векторів $\vec{x}_i$, $i = 1,\ldots,k$ належать множині цілих чисел. Для цього варіанту вектори $\vec{x}_i$, $i = 1,\ldots,k$ можуть мати тільки компоненти 0 або ±1. Вважаючи, що вектори $\vec{x}_i$ є ортонормованими, кожний з них повинен мати тільки одну компоненту ±1;

– (N) компоненти векторів $\vec{x}_i$, $i = 1,\ldots,k$ належать множині натуральних чисел, тобто вектори $\vec{x}_i$, $i = 1,\ldots,k$ можуть мати тільки компоненти 0 або 1. Так само як у варіанті (Z), вектор $\vec{x}_i$ повинен мати тільки одну компоненту рівну одиниці.

Таким чином варіанти (Z) та (N) задаються тільки діагональними коефіцієнтами матриць A_i і не залежать від коефіцієнтів, які не належать до діагональних коефіцієнтів. Їхня відмінність полягає у кількості глобальних екстремумів, тобто якщо у варіанта (N) кількість глобальних екстремумів рівна L, то для варіанта (Z) їх буде $2^k \cdot L$.

Розглянемо квадратичні екстремальні задачі, що відповідають кожному із зазначених трьох варіантів. Для кожного $i=1,\ldots,k$ визначимо коефіцієнти симетричної матриці A_i розміру $n\times n$ через a_{ijl}, $j=1,\ldots,n$, $l=1,\ldots,n$, а компоненти n-вимірного вектора $\vec{x}_i$ через x_{ij}, $j=1,\ldots,n$.

Квадратична екстремальна задача для варіанта (R) має такий вигляд:

$$Q_1^* = \min \sum_{i=1}^{k}\sum_{j=1}^{n}\sum_{l=1}^{n} a_{ijl}x_{ij}x_{il} \tag{15}$$

при обмеженнях:

$$\sum_{j=1}^{n} x_{ij}^2 \in 1,\ i=\overline{1,k}, \tag{16}$$

$$\sum_{l=1}^{n} x_{il}x_{jl} = 0, \quad i=\overline{1,k-1}, \quad j=\overline{i+1,k}. \tag{17}$$

Обмеження (16) задає умову нормованості векторів $\vec{x}_i$, $i=1,\ldots,k$, а обмеження (17) – умову їх взаємної ортогональності.

Опису задачі (15) – (17) відповідає наступний AMPL код:

```
# Опис параметрів задачі (15)-(17)
param n integer; # вимірність простору
param k <= n integer; # кількість векторів
param A{1..k, 1..n, 1..n};

# Опис змінних задачі (15)-(17)
var x{1..k, 1..n}; # для варіанту (R)
# var x{1..k, 1..n} integer; # для варіанту (Z)
# var x{1..k, 1..n} binary; # для варіанту (N)

# Опис задачі: цільова функція (15) та обмеження (16)-(17)
minimize obj15: sum{i in 1..k, j in 1..n, l in 1..n} (A[i,j,l] * x[i,j] *
x[i,l]);
subject to con16 {i in 1..k}: sum{j in 1..n} x[i,j] * x[i,j] = 1;
subject to con17 {i in 1..k-1, j in {(i + 1)..k}}: sum{l in 1..n} x[i, l] *
x[j, l]=0;
```

Щоб отримати кожен із трьох варіантів задачі (12) – (14) потрібно в AMPL коді вибрати той оператор опису змінних `x{1..k, 1..n}`, який відповідає дійсним (R), цілим (Z) чи натуральним (N) числам, які визначають компоненти векторів $\vec{x}_i$, $i=1,\ldots,k$.

Розглянемо декілька прикладів задачі (15) – (17).

Приклад 1. Розглянемо задачу (15) – (17) для матриць:

$$A_1 = diag(-1,2,3),\ A_2 = diag(4,-5,6),\ A_3 = diag(7,8,-9).$$

Для прикладу 1 всі точки глобального оптимуму легко визначити. Обмеження (16) – (17) означають, що x_i, $(i=1,2,3)$ знаходяться на одиничній сфері, звідки випливає, що $0\le x_{ij}^2\le 1$, для всіх $1\le i,j\le n$. Тому множина допустимих розв'язків складається з $8=2^3$ точок наступного вигляду:

$$x^* = \left((\pm 1,0,0)^T,(0,1,0)^T,(0,0,1)^T\right).$$

Оптимум цільової функції (15) рівний сумі від'ємних компонент матриць A_i, тобто для варіантів (R) та (Z) маємо $Q_1^* = Q_2^* = -15$. Для варіанта (N) також $Q_3^* = -15$, однак розв'язком буде єдина точка

$$x^* = \left((1,0,0)^T, (0,1,0)^T, (0,0,1)^T\right).$$

Вхідні дані для тестового прикладу на AMPL та запуск солвера BARON мають наступний вигляд:

```
data;
param n := 3;
param k := 3;
param A :=
[1,*,*]: 1 2 3 :=
1 -1 0 0
2  0 2 0
3  0 0 3
[2,*,*]: 1 2 3 :=
1  4 0 0
2  0 -5 0
3  0 0 6
[3,*,*]: 1 2 3 :=
1  7 0 0
2  0 8 0
3  0 0 -9
;
option baron_options 'barstats outlev=1 maxtime=86400';
solve;
display obj15;
display x;
```

У результаті обчислень за допомогою солвера BARON з NEOS сервера отримуємо наступний результат для варіанта (R):

```
************************************************************
   NEOS Server Version 6.0
   Solver    : go:BARON:AMPL
   Start     : 2022-07-17 08:37:13
   End       : 2022-07-17 08:37:15
   Host      : prod-sub-1.neos-server.org
************************************************************
You are using the solver baron.
Checking ampl.mod for baron_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-1.neos-server.org

9 variables, all nonlinear
6 constraints, all nonlinear; 27 nonzeros
      6 equality constraints
1 nonlinear objective; 9 nonzeros.

BARON 21.1.13 (2021.01.13): barstats
outlev=1
maxtime=86400
threads=4
```

```
================================================================================
 Doing local search
 Solving bounding LP
 Starting multi-start local search
 Preprocessing found feasible solution with value -15.0000000000
 Problem solved during preprocessing
 Lower bound is -15.0000000000

 Calculating duals

                        *** Normal completion ***

 Wall clock time:                     0.10
 Total CPU time used:                 0.10

 Total no. of BaR iterations:      -1
 Best solution found at node:      -1
 Max. no. of nodes in memory:       0

 All done
================================================================================
BARON 21.1.13 (2021.01.13): 0 iterations, optimal within tolerances.
Objective -15
Objective lower bound = -15, upper bound = -15
barstatus = 1, modelstatus = 1
max nodes in memory = 0
optimum found at node -1
Baron run time (excluding setup) = 0.09 seconds
obj15 = -15

x :=
1 1   -1
1 2    0
1 3    0
2 1    0
2 2   -1
2 3    0
3 1    0
3 2    0
3 3   -1
;
```

Покажемо, як за допомогою солвера BARON отримати розв'язок задачі для варіанта (N) та довести, що він є єдиним: Для цього розв'яжемо задачу (15) – (17) і після розрахунку отримаємо оптимальний розв'язок $x_1 = diag(1,1,1)$. Модифікуємо AMPL код, додавши до задачі (15) – (17) наступне обмеження:

```
subject to sol1: sum{i in 1..k, j in 1..n} (x[i,j]-x1[i,j])*(x[i,j]-x1[i,j]) >= 1
```

та знайдемо оптимальний розв'язок нової задачі. Для нього значення цільової функції дорівнює -3 та є більшим за -15, що свідчить про неможливість знаходження ще одного оптимального розв'язку, що відрізняється від уже знайденого.

За аналогічною схемою доведемо, що для варіанта (Z) задача (15) – (17) має 8 розв'язків. Запустимо AMPL код і після розрахунку отримаємо перший оптимальний розв'язок $x_1 = diag(-1,-1,-1)$. Додамо до AMPL коду наступне обмеження:

```
subject to sol1: sum{i in 1..k, j in 1..n} (x[i,j]-x1[i,j])*(x[i,j]-x1[i,j]) >= 1.
```

Доповнення задачі 1 заданим обмеженням означає те, що на наступних кроках солвер шукатиме оптимальний розв'язок, який відрізняється від уже знайденого.

Отримуємо другий оптимальний розв'язок $x_2 = diag(1,1,1)$, додамо до задачі наступне обмеження:

```
subject to sol2: sum{i in 1..k, j in 1..n} (x[i,j]-x2[i,j])*(x[i,j]-x2[i,j]) >= 1;
```

На третьому кроці – $x_3 = diag(1,-1,1)$, додамо обмеження:

```
subject to sol3: sum{i in 1..k, j in 1..n} (x[i,j]-x3[i,j])*(x[i,j]-x3[i,j]) >= 1;
```

На четвертому кроці – $x_4 = diag(-1,-1,1)$, додамо обмеження:

```
subject to sol4: sum{i in 1..k, j in 1..n} (x[i,j]-x4[i,j])*(x[i,j]-x4[i,j]) >= 1;
```

На п'ятому кроці – $x_5 = diag(-1,1,1)$, додамо обмеження:

```
subject to sol5: sum{i in 1..k, j in 1..n} (x[i,j]-x5[i,j])*(x[i,j]-x5[i,j]) >= 1;
```

На шостому кроці – $x_6 = diag(1,1,-1)$, додамо обмеження:

```
subject to sol6: sum{i in 1..k, j in 1..n} (x[i,j]-x6[i,j])*(x[i,j]-x6[i,j]) >= 1;
```

На сьомому кроці – $x_7 = diag(1,-1,-1)$, додамо обмеження:

```
subject to sol7: sum{i in 1..k, j in 1..n} (x[i,j]-x7[i,j])*(x[i,j]-x7[i,j]) >= 1;
```

На восьмому кроці – $x_8 = diag(-1,1,-1)$, додамо обмеження:

```
subject to sol8: sum{i in 1..k, j in 1..n} (x[i,j]-x8[i,j])*(x[i,j]-x8[i,j]) >= 1.
```

Знаходячи розв'язок задачі з вісьмома обмеженнями, отримуємо значення цільової функції, яке дорівнює -3. Воно є більшим за -15, що свідчить про неможливість знаходження 9-го оптимального розв'язку, що відрізняється від уже знайдених.

Приклад 2. Розглянемо задачу (15) – (17) для матриць:

$$A_1 = diag(1,2,3),\ A_2 = diag(4,5,6),\ A_3 = diag(7,8,9).$$

Вхідні дані для тестового прикладу та запуск солвера BARON мають наступний вигляд:

```
data;
param n := 3;
param k := 3;
param A :=
[1,*,*]: 1 2 3 :=
1  1 0 0
2  0 2 0
3  0 0 3
[2,*,*]: 1 2 3 :=
1  4 0 0
2  0 5 0
3  0 0 6
[3,*,*]: 1 2 3 :=
1  7 0 0
2  0 8 0
3  0 0 9
;
option baron_options 'barstats outlev=1 maxtime=86400';
solve;
display obj15;
display x;
```

Після завершення обчислень на NEOS сервері для варіанта (R) отримуємо наступний результат:

```
************************************************************
   NEOS Server Version 6.0
   Solver   : go:BARON:AMPL
   Start    : 2022-07-04 06:49:14
   End      : 2022-07-04 14:36:29
   Host     : prod-sub-1.neos-server.org
************************************************************
You are using the solver baron.
Checking ampl.mod for baron_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-5.neos-server.org

9 variables, all nonlinear
6 constraints, all nonlinear; 27 nonzeros
      6 equality constraints
1 nonlinear objective; 9 nonzeros.

BARON 21.1.13 (2021.01.13): barstats
outlev=1
maxtime=28000
threads=4
================================================================================
 Doing local search
 Solving bounding LP
 Starting multi-start local search
 Preprocessing found feasible solution with value  15.0000000000
 Done with local search
================================================================================
  Iteration    Open nodes         Time (s)    Lower bound      Upper bound
          1             1             0.12     12.0000          15.0000
*        39            18             0.36     12.0000          15.0000
*       295            37             1.34     12.0000          14.9999
*       494            63             2.16     12.0000          14.9999
*      1146           158             4.88     12.0022          14.9999
*      1906           255             8.03     12.0023          14.9999
                                        .
                                        .
                                        .
    5926435        353420         28000.00     14.9024          14.9998

                    *** Max. allowable time exceeded ***

 Wall clock time:                28016.54
 Total CPU time used:            28000.00

 Total no. of BaR iterations: 5926435
 Best solution found at node: 4689474
 Max. no. of nodes in memory:  353425

 All done
================================================================================
BARON 21.1.13 (2021.01.13): 5926435 iterations, CPU time limit reached.
Objective 14.9998337
```

```
Objective lower bound = 14.9023752058, upper bound = 14.9998337029
barstatus = 4, modelstatus = 4
max nodes in memory = 353425
optimum found at node 4689474
Baron run time (excluding setup) = 28000 seconds
obj15 = 14.9998

x :=
1 1    0.837889
1 2   -0.506838
1 3    0.202600
2 1    0.470505
2 2    0.482483
2 3   -0.738800
3 1    0.276711
3 2    0.714360
3 3    0.642736
;
```

З наведеного протоколу видно, що солвер BARON затратив 5926435 ітерацій за час, що становить 28000 секунд. Такий час є максимальним, що відводиться на розв'язання задачі на NEOS сервері. Варто зазначити, що у ході препроцесингу солвер BARON знайшов допустимий розв'язок зі значенням 15.0000. Однак, за відведений час солвер зміг покращити нижню границю для цільової функції лише до значення 14.9024.

Для варіанта (Z) солвер BARON отримав наступний результат:

```
************************************************************
   NEOS Server Version 6.0
   Solver   : go:BARON:AMPL
   Start    : 2022-07-18 11:37:18
   End      : 2022-07-18 11:37:20
   Host     : prod-sub-1.neos-server.org
************************************************************
You are using the solver baron.
Checking ampl.mod for baron_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-2.neos-server.org

9 variables, all nonlinear
6 constraints, all nonlinear; 27 nonzeros
      6 equality constraints
1 nonlinear objective; 9 nonzeros.

BARON 21.1.13 (2021.01.13): barstats
outlev=1
maxtime=86400
threads=4
================================================================================
 Doing local search
 Solving bounding LP
 Starting multi-start local search
 Done with local search
================================================================================
  Iteration    Open nodes         Time (s)    Lower bound      Upper bound
          1             1             0.15     12.0000         0.100000E+52
*         5             3             0.33     12.0000          15.0000
         45             0             0.43     15.0000          15.0000
```

```
 Calculating duals

                        *** Normal completion ***

 Wall clock time:                     0.44
 Total CPU time used:                 0.43

 Total no. of BaR iterations:      45
 Best solution found at node:       5
 Max. no. of nodes in memory:       8

 All done
================================================================================
BARON 21.1.13 (2021.01.13): 45 iterations, optimal within tolerances.
Objective 15
Objective lower bound = 15, upper bound = 15
barstatus = 1, modelstatus = 1
max nodes in memory = 8
optimum found at node 5
Baron run time (excluding setup) = 0.43 seconds
obj15 = 15

x :=
1 1   0
1 2   0
1 3   1
2 1   1
2 2   0
2 3   0
3 1   0
3 2   1
3 3   0
;
```

Для цього прикладу $Q_1^* = Q_2^* = Q_3^* = 15$.

Дослідимо кількість розв'язків, які має задача для варіанта (N). На першому кроці, в результаті запуску AMPL коду, який наведено вище, отримаємо один із оптимальних розв'язків, який будемо вважати першим розв'язком $x_1 = \big((0,1,0),(1,0,0),(0,0,1)\big)$. На другому кроці додамо до AMPL коду наступне обмеження:

```
subject to sol1: sum{i in 1..k, j in 1..n} (x[i,j]-x1[i,j])*(x[i,j]-x1[i,j])
>= 1.
```

У результаті отримаємо другий оптимальний розв'язок $x_2 = \big((0,0,1),(0,1,0),(1,0,0)\big)$. На третьому, четвертому, п'ятому та шостому кроках додаємо до AMPL коду наступні обмеження:

```
subject to sol2: sum{i in 1..k, j in 1..n} (x[i,j]-x2[i,j])*(x[i,j]-x2[i,j])
>= 1.
```

На третьому кроці – $x_3 = \big((1,0,0),(0,0,1),(0,1,0)\big)$, додамо обмеження:

```
subject to sol3: sum{i in 1..k, j in 1..n} (x[i,j]-x3[i,j])*(x[i,j]-x3[i,j])
>= 1.
```

На четвертому кроці – $x_4 = \big((0,0,1),(1,0,0),(0,1,0)\big)$, додамо обмеження:

```
subject to sol4: sum{i in 1..k, j in 1..n} (x[i,j]-x4[i,j])*(x[i,j]-x4[i,j])
>= 1.
```

На п'ятому кроці – $x_5 = \big((0,1,0),(0,0,1),(1,0,0)\big)$, додамо обмеження:

```
subject to sol5: sum{i in 1..k, j in 1..n} (x[i,j]-x5[i,j])*(x[i,j]-x5[i,j])
>= 1.
```

На шостому кроці – $x_6 = \big((1,0,0),(0,1,0),(0,0,1)\big)$, додамо обмеження:

```
subject to sol6: sum{i in 1..k, j in 1..n} (x[i,j]-x6[i,j])*(x[i,j]-
x6[i,j]) >= 1.
```

При подальших спробах розв'язання на сервері будемо отримувати від сервера повідомлення: "Problem is infeasible", що свідчить про неможливість знаходження 7-го оптимального розв'язку. Це доводить, що варіант (N) для 2 тесту має 6 оптимальних розв'язків.

Якщо у варіанта (N) кількість оптимальних розв'язків дорівнює 6, тоді для варіанта (Z) кількість оптимальних розв'язків буде рівною $2^3 \cdot 6 = 48$.

Приклад 3. Розглянемо задачу (15) – (17) для матриць

$$A_1 = \begin{pmatrix} 3 & 3.5 & -2 \\ 3.5 & 6 & -9 \\ -2 & -9 & 4 \end{pmatrix}, \; A_2 = \begin{pmatrix} -3 & -3 & 3.5 \\ -3 & 5 & -6 \\ 3.5 & -6 & 3 \end{pmatrix}.$$

Вхідні дані для тестового прикладу на AMPL та запуск солвера BARON мають наступний вигляд:

```
data;
param n := 3;
param k := 2;
param A :=
[1,*,*]: 1 2 3 :=
1  3 3.5 -2
2  3.5 6 -9
3  -2 -9  4
[2,*,*]: 1 2 3 :=
1  -3 -3 3.5
2  -3  5  -6
3  3.5 -6 3
;

option baron_options 'barstats outlev=1 maxtime=86400';
solve;
display obj15;
display x;
```

Після завершення обчислень за допомогою солвера BARON отримуємо наступний результат для варіанта (R):

```
************************************************************
   NEOS Server Version 6.0
   Solver    : go:BARON:AMPL
   Start     : 2022-07-17 08:34:45
   End       : 2022-07-17 08:34:53
   Host      : prod-sub-1.neos-server.org
************************************************************
You are using the solver baron.
Checking ampl.mod for baron_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-1.neos-server.org
```

```
6 variables, all nonlinear
3 constraints, all nonlinear; 12 nonzeros
      3 equality constraints
1 nonlinear objective; 6 nonzeros.

BARON 21.1.13 (2021.01.13): barstats
outlev=1
maxtime=86400
threads=4
================================================================================
 Doing local search
 Solving bounding LP
 Starting multi-start local search
 Preprocessing found feasible solution with value -8.50412496883
 Done with local search
================================================================================
  Iteration    Open nodes         Time (s)    Lower bound      Upper bound
          1             1             0.11    -18.0409         -8.50412
       1991             0             6.30    -8.50414         -8.50413

 Calculating duals
                         *** Normal completion ***

 Wall clock time:                     6.30
 Total CPU time used:                 6.30

 Total no. of BaR iterations:    1991
 Best solution found at node:    1658
 Max. no. of nodes in memory:      23

 All done
================================================================================
BARON 21.1.13 (2021.01.13): 1991 iterations, optimal within tolerances.
Objective -8.504131033
Objective lower bound = -8.50413953737, upper bound = -8.50413103323
barstatus = 1, modelstatus = 1
max nodes in memory = 23
optimum found at node 1658
Baron run time (excluding setup) = 6.29 seconds
obj15 = -8.50413

x :=
1 1   -0.0305674
1 2    0.680533
1 3    0.73208
2 1   -0.95314
2 2   -0.240397
2 3    0.183672
;
```

З наведеного протоколу видно, що на розв'язання задачі (15) – (17) для третього прикладу солвер BARON затратив 1991 ітерацій за час, що становить 6.29 секунд.

Висновки. У роботі досліджено структуру NEOS сервера, принципи його взаємодії з користувачем, доступні солвери та статистику надсилання робіт. При розв'язанні та дослідженні оптимізаційних задач на сервері найпопулярнішими форматами вхідних даних є GAMS та AMPL.

Було досліджено відому комбінаторну NP-повну задачу комівояжера та її розширення на випадок *m*-комівояжерів. Знайдено оптимальні розв'язки для різної кількості комівояжерів та максимальної кількості міст, які вони можуть відвідати. Проведено порівняння солверів CPLEX та

Gurobi для випадку одного комівояжера залежно від вибраної моделі. Для більшості експериментів солвер Gurobi затратив менше часу, ніж солвер CPLEX.

Розглянуто багатоекстремальну квадратичну задачу на многовиді Штіфеля. Наведено різні варіанти формулювань задачі, для яких проведені розрахунки за допомогою NEOS сервера, використовуючи для глобальної оптимізації солвер BARON. Проведено розрахунки на трьох прикладах, наведені знайдені оптимальні розв'язки та обґрунтовано їх кількість.

Робота підтримана грантом Volkswagen Foundation (грант № 97775).

Список літератури

1. NEOS Solver. https://neos-server.org/ (звернення: 15.12.2022)
2. Dolan E. The NEOS Server 4.0 Administrative Guide, 2001. 41 p.
3. Moré J.J., Czyzyk J.J., Mesnier M.P. The Network-Enabled Optimization System (NEOS) Server. 1998. 13 p.
4. Gropp W., Moré J.J. Optimization Environments and the NEOS Server. 1997. 18 p.
5. Fourer R., Gay D., Kernighan B. AMPL, A Modeling Language for Mathematical Programming. Belmont: Duxburry Press, 2003. 517 p.
6. CPLEX Optimizer. High-performance mathematical programming solver for linear programming, mixed-integer programming and quadratic programming. https://www.ibm.com/analytics/cplex-optimizer (звернення: 15.12.2022)
7. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com/ (звернення: 15.12.2022)
8. Tawarmalani M., Sahinidis N.V. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*. 2005. **103** (2). P. 225–249.
9. Sahinidis N.V. BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs. User's manual. 2021.
10. Kilinc M., Sahinidis N.V. Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems in BARON. *Optimization Methods and Software*. 2018. 33. P. 540–562.
11. Miller C.E., Tucker A.W., Zemlin R.A. Integer programming formulation of travelling salesman problem. *J. ACM*. 1960. 3. P. 326–329.
12. Гамецкий А.Ф., Соломон Д.И. Исследование операций. Том II. Кишинэу, Эврика, 2008. 592 с.
13. Алексеева Е.В. Построение математических моделей целочисленного линейного программирования. Примеры и задачи: Учеб. пособие / Новосиб. гос. ун-т. Новосибирск, 2012. 131 с.
14. Стецюк П.И., Нуриев У.Г., Нуриева Ф.У. Новая модель целочисленного линейного программирования для задачи коммивояжера. Матеріали 7-ї Міжнародної наукової конференції "Математичне моделювання, оптимізація та інформаційні технології (ММОТІ-2021)", 15–19 листопада 2021 р. С. 319–325.
15. Стецюк П.І., Соломон Д.І., Григорак М.Ю. Задачі про найкоротші *k*-вершинні цикли та шляхи. *Кібернетика та комп'ютерні технології*. 2021. № 3. С. 15–33. https://doi.org/10.34229/2707-451X.21.3.2
16. TSPLIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (звернення: 15.12.2022)
17. Balogh J., Csendes T., Rapcsák T. Some Global Optimization Problems on Stiefel Manifold. *Journal of Global Optimization*. 2004. 30. P. 91–101.
18. Шор Н.З., Стецюк П.И., Березовский О.А. Двойственные оценки для оптимизационной задачи квадратичного типа на многообразии Штиффеля. *Теория оптимальных решений*. Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины. 2004. С. 3–10. http://dspace.nbuv.gov.ua/handle/123456789/84870

Одержано 13.12.2022

Біла Галина Дмитрівна,
кандидат фізико-математичних наук, науковий співробітник
відділу математичних методів дослідження операцій
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
bila.galyna@gmail.com

Корчинський Олександр Олегович,
студент третього курсу Ужгородського національного університету, Ужгород,
ddphyk@gmail.com

Стецюк Петро Іванович,
доктор фізико-математичних наук, завідувач відділу методів негладкої оптимізації
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
stetsyukp@gmail.com
https://orcid.org/0000-0003-4036-2543

Хом'як Ольга Миколаївна,
кандидат фізико-математичних наук, старший науковий співробітник
відділу математичних методів теорії надійності складних систем
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
khomiak.olha@gmail.com
https://orcid.org/0000-0002-5384-9070

Шеховцов Сергій Борисович,
кандидат технічних наук, доцент кафедри інформаційних технологій
Харківського національного університету радіоелектроніки, Харків.
serhii.shekhovtsov@nure.ua
https://orcid.org/0000-0003-2381-7999

УДК 519.85

Г.Д. Біла [1], О.О. Корчинський [2], П.І. Стецюк [1 *], О.М. Хом'як [1], С.Б. Шеховцов [3]

Використання NEOS-сервера для розв'язання двох класів оптимізаційних задач

[1] Інститут кібернетики імені В.М. Глушкова НАН України, Київ
[2] Ужгородський національний університет, Україна
[3] Харківський національний університет радіоелектроніки, Україна
[*] Листування: stetsyukp@gmail.com

NEOS-сервер надає безкоштовний доступ до бібліотеки необхідного програмного забезпечення для розв'язання оптимізаційних задач. Розроблені ефективні чисельні методи можуть бути використані для конкретного застосування або для вирішення широкого кола задач математичного програмування.

Стаття присвячена дослідженню можливостей розв'язання задач оптимізації за допомогою NEOS сервера на прикладі двох задач математичного програмування: задачі лінійного булевого програмування для відомої задачі m-комівояжерів та задачі нелінійного програмування для спеціальної багатоекстремальної задачі на многовиді Штіфеля.

Матеріал роботи викладений у 4 розділах. У розділі 1 описано загальні відомості про використання NEOS сервера для вирішення задач оптимізації та приділено основну увагу мові математичного програмування AMPL як його складовій частині. Проведено порівняння частоти використання AMPL, GAMS та інших форматів вхідних даних, які доступні на сервері.

У другому розділі описано принцип розв'язання задач оптимізації за допомогою NEOS та наведено список наявних на сервері солверів, що підтримують AMPL. Наведено статистику частоти використання солверів CPLEX, Gurobi та BARON у порівнянні з іншими доступними на сервері солверами, а також загальну статистику кількості розв'язаних задач на сервері за 2021 рік.

У розділі 3 описано задачі лінійного булевого програмування для задачі m-комівояжерів, яка при m=1 рівносильна відомій задачі комівояжера. Наведено показники ефективності розв'язання цих задач за допомогою солверів CPLEX та Gurobi.

У розділі 4 проведено дослідження задачі нелінійного програмування для спеціальної багатоекстремальної задачі на многовиді Штіфеля. Наведено загальне формулювання задачі та формулювання її часткових випадків у залежності від обмежень, накладених на компоненти векторів. Досліджено ефективність розв'язання за допомогою солвера BARON тестових прикладів залежно від кількості векторів та їх розмірності.

Ключові слова: оптимізація, NEOS сервер, AMPL, NEOS солвери, задача комівояжера, многовид Штіфеля.

UDC 519.85

Halyna Bila [1], Oleksandr Korchynskyy [2], Petro Stetsyuk [1 *], Olha Khomiak [1], Serhiy Shekhovtsov [3]

Using the NEOS Server for Solving Two Classes of Optimization Problems

[1] *V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv*
[2] *Uzhhorod National University, Ukraine*
[3] *Kharkiv National University of Radio Electronics, Ukraine*
[*] *Correspondence: stetsyukp@gmail.com*

NEOS server provides free access to the library of necessary software for solving optimization problems. The effective numerical methods developed can be used for a specific application or for solving a wide range of mathematical programming problems.

The article is devoted to researching the possibilities of solving optimization problems using NEOS server on the example of two mathematical programming problems: a linear Boolean programming problem for the well-known m-traveler problem and nonlinear programming problem for a special multiextremal problem on the Stiefel's manifold.

The material of the work is presented in 4 sections. Chapter 1 describes general information about using NEOS server for solving optimization problems and focuses on AMPL mathematical programming language as its component. Comparison of the frequency of use of AMPL, GAMS and other input data formats available on the server was conducted.

The second section describes the principle of solving optimization problems using NEOS and provides a list of AMPL-supporting solvers available on the server. Statistics of use frequency of the solvers CPLEX, Gurobi and BARON in comparison with other solvers available on the server, as well as general statistics of the number of solved problems on the server in 2021 are given.

Chapter 3 describes linear Boolean programming problems for the m-travelers problem, which is equivalent to the well-known traveling salesman problem. Performance indicators for solving these problems using CPLEX and Gurobi solvers are given.

In Chapter 4 a study of nonlinear programming problem for a special multiextremal problem on the Stiefel's manifold is conducted. General formulation of the problem and the formulation of its partial cases depending on the constraints imposed on vector components are given. The efficiency of solving test cases using BARON solver was investigated depending on the number of vectors and their dimensions.

Keywords: optimization, NEOS server, AMPL, NEOS solvers, travelling salesman problem, Stiefel's manifold.