УДК 338.45(078.5)

О. Г. Додонов[1], А. І. Кузьмичов[2]
[1]Інститут проблем реєстрації інформації НАН України
вул. М. Шпака, 2, 03113 Київ, Україна
e-mail: dssd@ipri.kiev.ua
[2]Академія муніципального управління МОН України
вул. І. Кудрі, 33, 01601 Київ, Україна
e-mail: akuzmychov@gmail.com

Оптимізаційні моделі еволюційного програмування в Excel: розв'язання задачі комівояжера з обмеженнями alldifferent

Актуальність і масовий характер проблематики оптимізаційного моделювання привели до розробки та широкомасштабного впровадження доступних, потужних й ефективних засобів математичної оптимізації. Вважається, що цей процес має привести до суттєвого підвищення якості управлінських рішень, що приймаються на різних рівнях відповідальності. Важливо, що ці засоби орієнтовані на розв'язання задач оптимізації, реалізація математичних моделей яких до сьогодні була недосяжна для звичайних управлінців та аналітиків. Розглянуто новітні програмні засоби еволюційного програмування, які дозволяють на звичайному робочому місці професійного користувача без будь-яких додаткових витрат розв'язувати оптимізаційну задачу комівояжера, яка має безліч варіантів і модифікацій.

***Ключові слова**: математична оптимізація, MS Excel, еволюційне програмування, генетичний алгоритм, задача комівояжера, прийняття рішень, spreadsheet modeling and optimization.*

Вступ

Переважну більшість практичних задач оптимізації відносять до «важких» для комп'ютерної реалізації задач з-за явної нелінійності цільової функції й обмежень і необхідності врахування спеціальних граничних значень для шуканих невідомих. Тож вимушений й тому найбільш популярний шлях до пошуку глобального оптимуму — зведення цих задач до класичних моделей лінійного, цілочислового або хоча б опуклого програмування, зрозуміло, із втратою певних характерних властивостей об'єкта дослідження. Цей шлях фактично досяг вершин своїх теоретичних і прикладних можливостей, відповідно, залишається актуальною

проблема безпосереднього математичного моделювання реальних задач оптимізації без будь-яких принципових спрощень. І у цьому напрямку йде пошук оригінальних обчислювальних алгоритмів, за допомогою яких із застосуванням потужних комп'ютерних засобів вдається знайти оптимальні чи близькі до них розв'язки важливих і «важких» задач солідних розмірів із явним практичним спрямуванням. Цей напрямок орієнтується на комбіновану технологію імітаційної оптимізації (математичне програмування + імітаційне моделювання, simulation optimization), де спільно застосовується апарат обробки невідомих детермінованої й випадкової природи.

Однією з таких «важких» задач, яка має безліч застосувань, зокрема, як тест (benchmark) для дослідження нових алгоритмів, є класична задача про комівояжера (який має відвідати n міст з мінімальною довжиною контуру обходу), де кінцевий результат критично й експоненційно залежить від розміру задачі (значення n) — кількість можливих контурів обходу складає фантастичну величину $n!$ У процесі її розв'язання типовими й паралельними алгоритмами переборного типу протягом останніх майже 60 років сформувався навіть стимул спортивного типу у вигляді досягнення рекордів — час від часу дослідники публікують кращі показники щодо досягнутих максимальних розмірів (значень n), витрат машинного часу спеціально побудованих багатомашинних комп'ютерних систем і поведінки відповідних алгоритмів. Найперший рекорд, який було досягнуто ще на ЕОМ ранніх поколінь, це найкоротший контур, що з'єднує 49 міст-столиць штатів США. Останні рекорди 2000-х років, що отримані на багатомашинних комплексах за спеціально розробленою програмою, досить вражаючі: 2005 р. — 33810 міст, 2006 р. — 85900 міст[1]. Але ці рекорди, на жаль, не мають серйозного впливу на реальну масову практику, де сотні тисяч аналітиків мають, як правило, стандартні комп'ютерні та програмні засоби для моделювання практичних задач оптимізації й прийняття управлінських рішень на цій основі.

Тож саме для цієї найпоширенішої категорії користувачів-оптимізаторів зі стандартною математичною й програмістською підготовкою вже 20 років у популярне обчислювальне середовище Excel вбудовують доступні й ефективні програмні продукти масового використання у вигляді готових до використання оптимізаційних програм-надбудов (solvers). Відповідно, у сфері оптимізаційного моделювання та прийняття управлінських рішень, яка найбільш поширена у бізнес-середовищі, в операційному менеджменті та бізнес-освіті, сформувалася й активно розвивається сучасна технологія аналізу, моделювання та прийняття рішень на платформі електронних таблиць (spreadsheets) — *електронно-табличне моделювання та оптимізація* (spreadsheet-based modeling and optimization) [4–5].

Стандартний комплект надбудови під назвою Excel Solver (в російськомовній версії — Поиск решения) до версії Excel 2003 складається з трьох програм-оптимізаторів для розв'язання найпоширеніших задач математичного програмування: лінійної, цілочислової та гладкої нелінійної оптимізації, де реалізовано три

[1] Слід згадати піонерні дослідження київських кібернетиків 70–80 років, коли в умовах обмежених машинних ресурсів цифрових ЕОМ (швидкодія, пам'ять) розроблялися спеціалізовані електронні моделі потокових задач, у тому числі, задачі комівояжера, де процес паралельних обчислень здійснюється зі швидкістю руху електричного струму, тобто, миттєво [2].

обчислювальні алгоритми: симплекс-метод, метод гілок і границь та методи Ньютона й спряжених градієнтів для нелінійних задач.

За узгодженням з компанією Microsoft до середовищ Excel 2007/2010 вбудовано вдосконалену надбудову Premium Solver, де для гладкої нелінійної оптимізації застосовується потужний метод узагальненого приведеного градієнта[2] (Generalized Reduced Gradient, GRG).

У прикладній математиці «важкі» задачі оптимізації відносять до класу нелінійних задач із негладкою цільовою функцією й обмеженнями на невідомі різного типу, для яких безсилі класичні моделі й методи нелінійного програмування градієнтного типу щодо пошуку глобального оптимуму. Тож прогресивна й багатообіцяюча надбудова Premium Solver має у своєму складі модуль негладкої нелінійної оптимізації під назвою Evolutionary Solver (у російськомовній версії Excel 2010 — Эволюционный поиск решения), що відповідає новітньому класу математичних моделей еволюційного програмування. В основі цього потужного, але мало ще відомого масовому користувачу модулю реалізація специфічного алгоритму направленого перебору під назвою «генетичний алгоритм» (ГА), де спільно застосовується генератор випадкових чисел й оригінальний обчислювальний механізм пошуку оптимуму, що відтворює еволюційні процеси живої природи. На відміну від стихійних методів спроб і помилок чи випадкового пошуку із накопиченням статистики, в ГА на кожному кроці-спробі здійснюється певне перекомбінування значень шуканих невідомих (мовою біології — мутація та схрещення) й розв'язується набір проміжних підзадач, щоб сформувати оновлений набір значень шуканих невідомих, який визначає краще за попереднє значення цільової функції. За цим алгоритмом можна знайти досить близький до оптимального розв'язок поставленої негладкої нелінійної задачі оптимізації солідного розміру [7, 8].

Цей же модуль має ще одну цікаву й ефективну властивість, що детально розглядається у цій роботі, яка дозволяє застосувати для шуканих невідомих оригінальний вид обмежень під назвою alldifferent (в Excel 2010 — «все разные»).

Для задачі про комівояжера цей вид обмежень є надзвичайно корисним, бо приводить до суттєвого збільшення допустимого розміру задачі, що розв'язується в Excel, — замість задачі з $n^2 + n$ невідомими за цим обмеженням розв'язується задача лише з n невідомими. Скажімо, за допомогою методу «гілок і границь» в Excel можна точно розв'язати задачу комівояжера за моделлю булевого програмування з $n \le 13$ [6, с. 192], зате еволюційним методом з обмеженням «все разные» наближено, з відхиленням від оптимуму на 2–3 %, що для практики цілком прийнятно, розв'язується задача комівояжера з $n \le 200$[3].

Появу надбудови Premium Solver слід вважати революційною подією у сфері прикладного оптимізаційного моделювання завдяки наявності доступних, потужних, ефективних й надзвичайно корисних засобів. На прикладі розв'язання класичної й «непідйомної» задачі комівояжера рядовий користувач (менеджер-аналітик, дослідник-практик, викладач, аспірант, студент і навіть учень старших кла-

[2] В Excel 2010 буквальний переклад — «Нелинейный метод обобщенного понижающего градиента (ОПГ)».
[3] Число 200 — максимальне граничне значення числа невідомих у стандартних версіях Excel у складі пакета MS Office, у комерційних версіях — кілька тисяч.

сів) фактично вперше на звичайному ПК із встановленим процесором електронних таблиць Excel 2007/2010 отримав реальну можливість перейти від «іграшкових» задач про комівояжера з приблизно 10–12 містами для обходу й тривалістю пошуку до 15–20 хв. до розв'язання цієї ж задачі на порядок більших розмірів із такою ж тривалістю, яка зазвичай покриває реальні задачі, що виникають у практиці моделювання матеріальних чи фінансових потоків у мережах та в прикладній логістиці.

Задача комівояжера

Задача комівояжера (в теорії графів — задача пошуку найкоротшого гамільтонова контуру) дуже близька за своєю постановкою до типових потокових задач на графах: про найкоротший шлях, про призначення та про мінімальне покриття. Але ці задачі розв'язуються значно простіше, ніж задача комівояжера, тому й розв'язок задачі комівояжера базується на алгоритмах цих дещо простіших задач.

Уперше математична модель задачі про комівояжера (TSP, travelling salesmam problem) була визначена в 1930 р. (К. Менгер) у класі моделей комбінаторного програмування; для її комп'ютерної реалізації у 1950-ті роки (Дж. Данциг) була поставлена відповідна задача лінійного програмування[4] (ЛП) із-за її схожості з транспортною задачею та задачею про призначення матричного типу. Правда, зразу ж виявилася непередбачувана проблема — при використанні лінійної моделі для задачі про призначення шуканий гамільтонів контур розривається на сукупність часткових підконтурів, що з'єднують певні групи точок. Ця постановка для повного графа з n вершинами містить $n^2 + n$ змінних і не менше половини цієї величини — це явні обмеження на невідомі, наступні неявні обмеження в процесі обчислень ставлять за мету ліквідацію часткових контурів. Тож хоча принципово симплекс-методом можна користуватися, але на практиці від нього відмовляються з-за його громіздкості й неефективності моделі ЛП. В Excel застосовується вбудований в «Поиск решения» точний симплекс-метод (див. нижче).

Наступний етап розробки алгоритмів для розв'язання задачі про комівояжера — більш продуктивний дискретний підхід з використанням розгалужень, алгоритмів направленого й скороченого перебору варіантів (адже повний перебір астрономічної $n!$ кількості варіантів відпадає одразу) за допомогою дерева пошуку та моделі булевого програмування. У цьому напрямку розробляються точні й наближені методи і алгоритми пошуку оптимуму. В Excel для цього застосовується вбудований в «Поиск решения» точний метод «гілок і границь»[5] (модель булевого програмування) і наближений еволюційний метод (модель еволюційного програмування).

Постановка задачі.

Комівояжер (агент роздрібної торгівлі) має обійти n клієнтів (замовників, певних об'єктів, традиційно — міст) найкоротшим шляхом (якнайшвидше, з мінімальними витратами ресурсів) і повернутися у початковий пункт, побувавши у кож-

[4] *Dantzig G.* Solution of a Large-Scale Traveling Salesman Problem / G. Dantzig, D. Fulkerson, S. Jonson. — OR, 1954. — P. 393–410.

[5] An Algorithm for Traveling Salesman Problem / [Little J., Murty K., Swenney D., Karel C.]. — OR, 1963. — P. 972–989.

ного клієнта лише раз. У ролі комівояжера може бути транспортний засіб з доставки товарів у заклади роздрібної торгівлі чи в певні пункти для поповнення запасів ресурсів (харчів, питної води, палива, ліків, джерел електроенергії) чи заміни персоналу, сміттєзбиральна техніка, кандидат в депутати, персонал контрольно-ревізійної служби, бригада обслуговування банкоматів, автобус з групою туристів тощо — усі вони мають на меті визначити оптимальний контур обходу відповідних об'єктів, щоб мінімізувати довжину, час чи інші витрати.

Аналогічну постановку має виробнича задача про переналагодження універсального обладнання для послідовного виконання на ньому n різних операцій із мінімальною загальною тривалістю усього процесу, задача про найкоротше з'єднання n електронних елементів на мікросхемі та багато інших.

Класична постановка задачі комівояжера доповнюється рядом специфічних чи узагальнених задач, коли, скажімо, комівояжер має відвідувати певні групи міст у межах певних регіонів чи коли треба визначити «вузьке місце» в контурі обходу — ці умови ще додатково ускладнюють й без цього непросту задачу в класичній постановці, але ж досягнення оптимального чи хоча би близького до нього варіанта забезпечує явну економічну чи іншу ефективність.

Ця математична задача припускає просту наочну інтерпретацію, з чого й розпочинається її постановка: на площині задано n точок із відповідними координатами, де кожна точка з'єднана із усіма іншими n точками. Це — набір даних (V, D, A, X, S), де

$V = \{1, \ldots, n\}$ — задана множина n вузлів (номери, назви, коди);

$D = \{(i, j)\}$ — задана множина n^2 дуг: (1, 1), (1, 2), …, (n, n), кожна (i, j)-та дуга ($i \neq j$) графа з'єднує певну пару вузлів $i, j \in V$, це — повний граф, для якого існує розв'язок; для неповного графа існування розв'язку не гарантується;

$A = \{a_{ij}\}$ — задана множина вагових коефіцієнтів чи параметрів дуг, це: відстань, тривалість, витрати палива, коштів чи будь-яких ресурсів, для $i = j$ $a_{ij} = \infty$; при $a_{ij} = a_{ji}$ задача симетрична, інакше, у загальному випадку, несиметрична;

$X = \{x_{ij}\}$ — шуканий результат у матричній формі, множина шуканих невідомих, де: $x_{ij} = 1$, якщо (i, j)-та дуга належить шуканому контуру, $x_{ij} = 0$ — в іншому випадку, або $X = \{x_i\}$ — шуканий результат у векторній формі, множина шуканих невідомих у вигляді комбінації n номерів вузлів;

S — схема контуру на площині.

Відповідно, для постановки та розв'язання задачі про комівояжера маємо зважений граф (або неорієнтовану мережу), де треба знайти найкоротший замкнений контур, який складається точно з n дуг і проходить через кожен вузол один лише раз (це задача про мінімальний гамільтонів контур, що визначена ірландським математиком Гамільтоном у 1800-ті роки, яка довгі роки сприймалася як звичайна головоломка).

Задача оптимізації.

I. Знайти план $X = \{x_{ij}\}$ та $U = \{u_i\}$, такі, щоб

II. ЦФ $K = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij} \to \min$

III. за обмежень:

а) $\sum_{i=1}^{n} x_{ij} = 1$, з кожного вузла можна вийти 1 раз;

б) $\sum_{j=1}^{n} x_{ij} = 1$, у кожен вузол можна увійти 1 раз;

ці два обмеження забезпечують умову 1-разового відвідування кожного міста;

в) $u_i - u_j + nx_{ij} \le n-1, i, j = 2,3,...,n, i \ne j$, u_i, u_j — довільні значення, при $i = j$ $u_i - u_j + nx_{ij} = 0$; це обмеження забезпечує неможливість розриву контуру на окремі підконтури та граничних умов: усі $x_{ij} \in \{0,1\}$.

n	n^2+n-1	n^2+1
4	19	17
5	29	26
6	41	37
7	55	50
8	71	65
9	89	82
10	109	101
11	131	122
12	155	145
13	181	170
14	209	197
15	239	226

Таким чином, для задачі обходу n міст треба знайти: n^2 значень матриці X, $n-1$ значень вектора U, разом: $n^2 + n - 1$ невідомих із загальним числом обмежень $n^2 + 1$ (див. табл.). Отже, з допустимим числом невідомих (200) і обмежень (100) в Excel можна розв'язати задачу комівояжера з числом міст до 13.

Розв'язання задачі комівояжера в Excel

Приклад 1. Симплекс-метод лінійного програмування.

Постановка задачі.

На площині задано координати 6 пунктів, розв'язати задачу комівояжера як закриту задачу про призначення (кількість претендентів = кількості вакансій).

Математична модель:

I. Знайти план $X = \{x_{ij}\}$ такий, щоб

II. ЦФ $K = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} \to \min$

III. за обмежень:

а) $\sum_{i=1}^{n} x_{ij} = 1$, з кожного вузла можна вийти 1 раз;

б) $\sum_{j=1}^{n} x_{ij} = 1$, у кожен вузол можна увійти 1 раз;

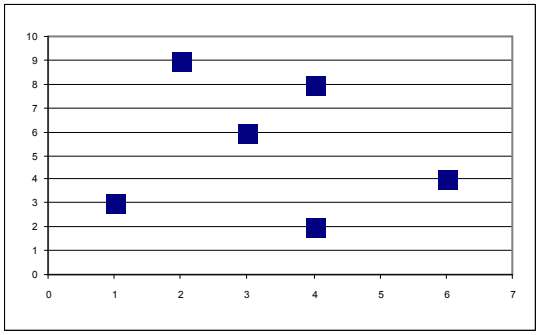
ці два обмеження забезпечують умову 1-разового відвідування кожного міста та граничних умов: усі $x_{ij} \in \{0,1\}$.

Порядок роботи.

1. Увести назви міст і їхні координати.

2. Побудувати діаграму Точечная:

M-1	3	6
M-2	4	2
M-3	6	4
M-4	2	9
M-5	4	8
M-6	1	3



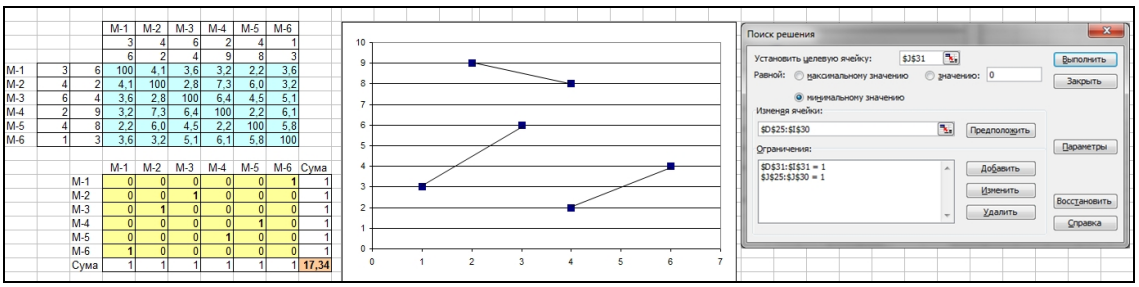
3. Сформувати матрицю відстаней A(6,6) обчисленнями значень a_{ij} за формулою: $a_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, усі діагональні елементи (a_{ii}) = 100 (аналог ∞):

			M-1	M-2	M-3	M-4	M-5	M-6
			3	4	6	2	4	1
			6	2	4	9	8	3
M-1	3	6	100	4,1	3,6	3,2	2,2	3,6
M-2	4	2	4,1	100	2,8	7,3	6,0	3,2
M-3	6	4	3,6	2,8	100	6,4	4,5	5,1
M-4	2	9	3,2	7,3	6,4	100	2,2	6,1
M-5	4	8	2,2	6,0	4,5	2,2	100	5,8
M-6	1	3	3,6	3,2	5,1	6,1	5,8	100

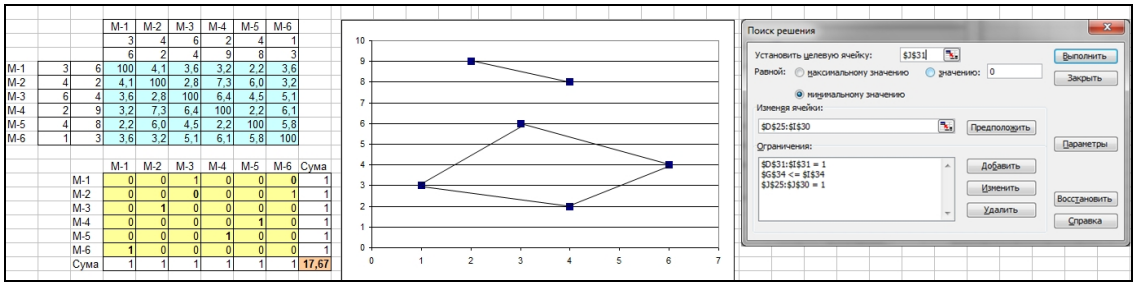
4. Сформувати матрицю *X*, знайти суми елементів за рядками та стовпцями й обчислити значення цільової функції (ЦФ):

	A	B	C	D	E	F	G	H	I	J
14				M-1	M-2	M-3	M-4	M-5	M-6	
15				3	4	6	2	4	1	
16				6	2	4	9	8	3	
17	M-1	3	6	100	4,1	3,6	3,2	2,2	3,6	
18	M-2	4	2	4,1	100	2,8	7,3	6,0	3,2	
19	M-3	6	4	3,6	2,8	100	6,4	4,5	5,1	
20	M-4	2	9	3,2	7,3	6,4	100	2,2	6,1	
21	M-5	4	8	2,2	6,0	4,5	2,2	100	5,8	
22	M-6	1	3	3,6	3,2	5,1	6,1	5,8	100	
23										
24				M-1	M-2	M-3	M-4	M-5	M-6	Сума
25			M-1							0
26			M-2							0
27			M-3							0
28			M-4							0
29			M-5							0
30			M-6							0
31			Сума	0	0	0	0	0	0	**0**

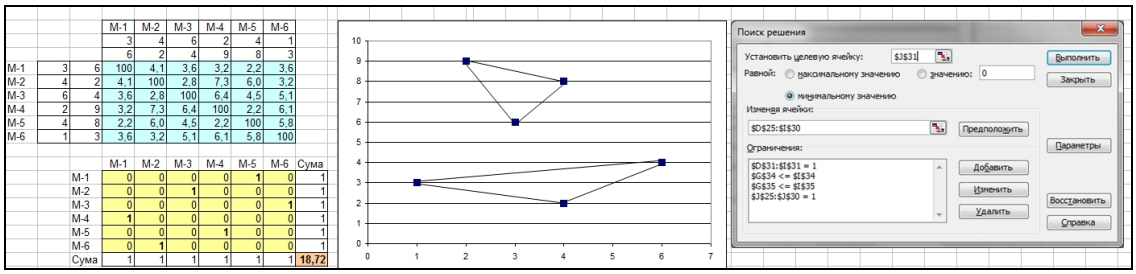
5. Сформувати табличну модель задачі про призначення й знайти її розв'язок. Як і передбачалося, шуканий контур розпався на три підконтури: (1-6-1), (2-3-2), (4-5-4), які побудовано на діаграмі, значення плану: (x_{16}, x_{23}, x_{32}, x_{45}, x_{54}, x_{61}) = 1, інші — нулі, ЦФ = 17,34:



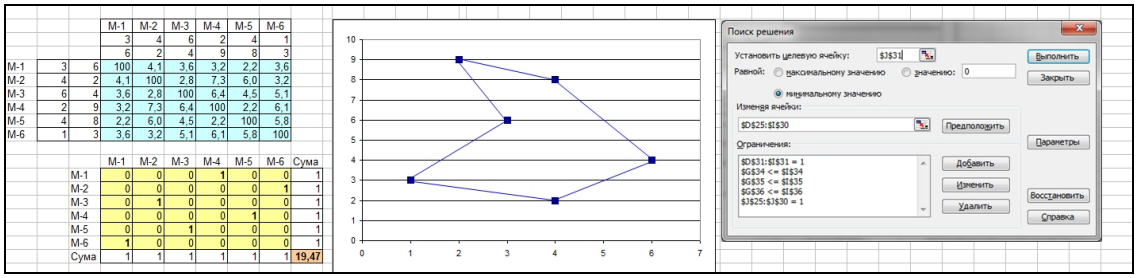
6. Вирішуємо розірвати підконтур (1-6-1), для цього формуємо додаткове обмеження: $x_{16} + x_{61} \le 1$ й уводимо в модель; отримано новий результат, за яким є два підконтури: (1-3-2-6-1) та (4-5-4), значення ЦФ збільшилося на 0,33:



7. Вирішуємо розірвати підконтур (4-5-4), для цього формуємо друге додаткове обмеження: $x_{45} + x_{54} \le 1$ і вводимо в модель; отримано новий результат, за яким є два підконтури: (1-5-4-1) та (2-3-6-2), значення ЦФ збільшилося на 1,05:



8. Вирішуємо розірвати підконтур (1-5-4-1), для цього формуємо третє додаткове обмеження: $x_{15} + x_{51} + x_{54} + x_{45} + x_{41} + x_{14} \le 2$ і вводимо його в модель; отримано новий результат, за яким знайдено оптимальний контур (1-4-5-3-2-6-1), його мінімальна довжина (значення ЦФ) дорівнює 19,47:



Висновок: модель лінійного програмування вимагає виконувати багато допоміжної роботи щодо визначення підконтурів для розірвання та уведення відповідних обмежень.

Приклад 2. Метод «гілок і границь» булевого програмування.
Постановка задачі (див. Приклад 1).
Використовується наведена вище математична модель.

Порядок роботи.
Виконати пункти 1÷5.
6. Сформувати допоміжну матрицю з метою реалізації обмеження (в), використавши шаблон матриці *X*.

7. Визначити стовпець із 5 клітинок для елементів u_i шуканого вектора U, транспонуванням визначити рядок з 5 клітинок для елементів u_j.

8. Сформувати матрицю лівих частин обмежень, на її діагоналі нулі:

E36 =$J36-E$40+6*E27

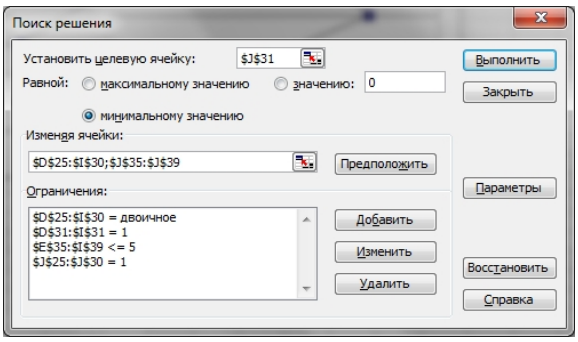
	A	B	C	D	E	F	G	H	I	J
33				M-1	M-2	M-3	M-4	M-5	M-6	
34			M-1							Ui
35			M-2		0	5	-4	-2	1	-1
36			M-3		1	0	-3	5	2	0
37			M-4		4	3	0	2	5	3
38			M-5		2	1	4	0	3	1
39			M-6		5	-2	-5	-3	0	-2
40				Uj	-1	0	3	1	-2	

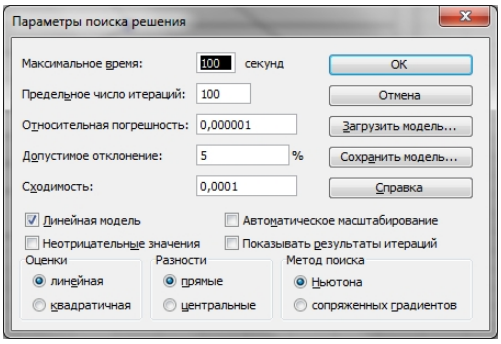
9. У табличну модель увести обмеження (в), для цього у вікні *Поиск решения*:

— у поле *Изменяя ячейки* додати адреси елементів u_i;

— у полі *Ограничения*: додати обмеження на двійковий тип матриці X та обмеження (в);

— у вікні *Параметры поиска решения* зняти галочку з позиції *Неотрицательные значения*, оскільки змінна u_i може приймати довільні значення.

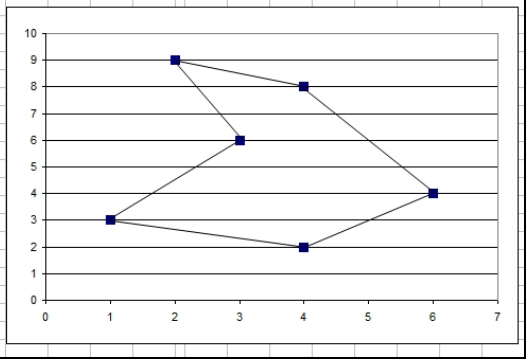




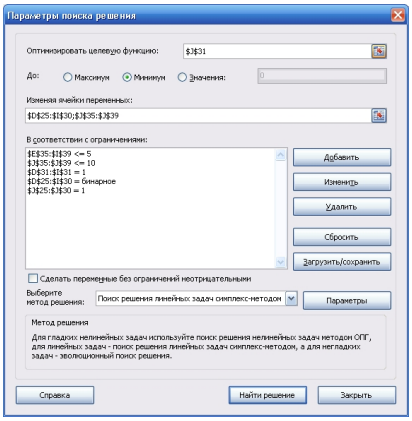
Результат:

E36 =$J36-E$40+6*E27

	A	B	C	D	E	F	G	H	I	J
14				M-1	M-2	M-3	M-4	M-5	M-6	
15				3	4	6	2	4	1	
16				6	2	4	9	8	3	
17	M-1	3	6	100	4,1	3,6	3,2	2,2	3,6	
18	M-2	4	2	4,1	100	2,8	7,3	6,0	3,2	
19	M-3	6	4	3,6	2,8	100	6,4	4,5	5,1	
20	M-4	2	9	3,2	7,3	6,4	100	2,2	6,1	
21	M-5	4	8	2,2	6,0	4,5	2,2	100	5,8	
22	M-6	1	3	3,6	3,2	5,1	6,1	5,8	100	
23										
24				M-1	M-2	M-3	M-4	M-5	M-6	Сума
25			M-1	0	0	0	0	0	1	1
26			M-2	0	0	1	0	0	0	1
27			M-3	0	0	0	0	1	0	1
28			M-4	1	0	0	0	0	0	1
29			M-5	0	0	0	1	0	0	1
30			M-6	0	1	0	0	0	0	1
31			Сума	1	1	1	1	1	1	19,47
32										
33				M-1	M-2	M-3	M-4	M-5	M-6	
34			M-1							Ui
35			M-2		0	5	-4	-2	1	-1
36			M-3		1	0	-3	5	2	0
37			M-4		4	3	0	2	5	3
38			M-5		2	1	4	0	3	1
39			M-6		5	-2	-5	-3	0	-2
40				Uj	-1	0	3	1	-2	



Висновок: цим методом можна знайти зразу шуканий контур обходу, але за цю можливість ми «розплачуємося» додатковими обмеженнями, тож область роботи цього методу суттєво обмежується граничними значеннями кількості невідомих та обмежень Excel: до 200 змінних і до 100 обмежень.



Зауваження. В середовищі Excel 2010 у вікно *Параметры поиска решения* треба внести такі зміни: для шуканих змінних u_i рекомендується внести певне обмеження, наприклад, не менше 10, далі у полі *Выберите метод решения* вибрати *Поиск решения линейных задач симплекс-методом*.

Приклад 3. Еволюційний метод.

Цю ж задачу комівояжера для найкоротшого обходу 6 пунктів розв'яжемо за допомогою методу ***Эволюционный поиск решения*** в Excel 2010.

Позначення:

i — поточний номер шуканої змінної (співпадає із номером міста в контурі), $i = 1, \ldots, 6$;

$X = \{x_i\}$ — шуканий вектор невідомих, $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$;

$D = \{d(i, j)\}$ — вектор відстаней між парами сусідніх вузлів у контурі обходу розміром з 6 елементів, де $i \prec j$ (i *передує* j).

Задача оптимізації:

I. Знайти план обходу $X = \{x_i\}$, такий, щоб

II. ЦФ: довжина контуру $K = \sum_{i,j=1}^{6} d_{ij} \to \min$

III. За обмежень: $x_i \in \{$***все разные***$\}$.

Порядок роботи.

Повторити пункти 1÷3.

4. Сформувати рядок X і заповнити його порядковими номерами, у наступній $(n + 1)$-й клітинці зробити посилання на 1-й елемент цього рядка (щоб замкнути контур або цикл). Обчислити відстані між сусідніми клітинками за допомогою функції ИНДЕКС й обчислити їхню суму за допомогою функції СУММ.

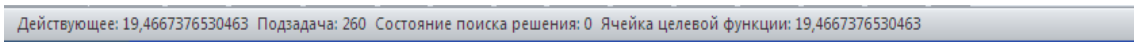
E54 =ИНДЕКС($D$46:$I$51;D53;E53)

	A	B	C	D	E	F	G	H	I	J	K
43				M-1	M-2	M-3	M-4	M-5	M-6		
44				3	4	6	2	4	1		
45				6	2	4	9	8	3		
46	M-1	3	6	100	4,1	3,6	3,2	2,2	3,6		
47	M-2	4	2	4,1	100	2,8	7,3	6,0	3,2		
48	M-3	6	4	3,6	2,8	100	6,4	4,5	5,1		
49	M-4	2	9	3,2	7,3	6,4	100	2,2	6,1		
50	M-5	4	8	2,2	6,0	4,5	2,2	100	5,8		
51	M-6	1	3	3,6	3,2	5,1	6,1	5,8	100		
52											
53			X=	1	2	3	4	5	6	1	ЦФ
54			Відстань=		4,12	2,83	6,40	2,24	5,83	61	25,03

5. Командою *Данные* → *Поиск решения* викликати вікно *Параметры поиска решения* і заповнити його поля вказаним чином.

6. Натиснути на *Параметры*, визначити у цьому вікні режим роботи програми, *ОК* і в попередньому вікні натиснути на *Найти решение*.

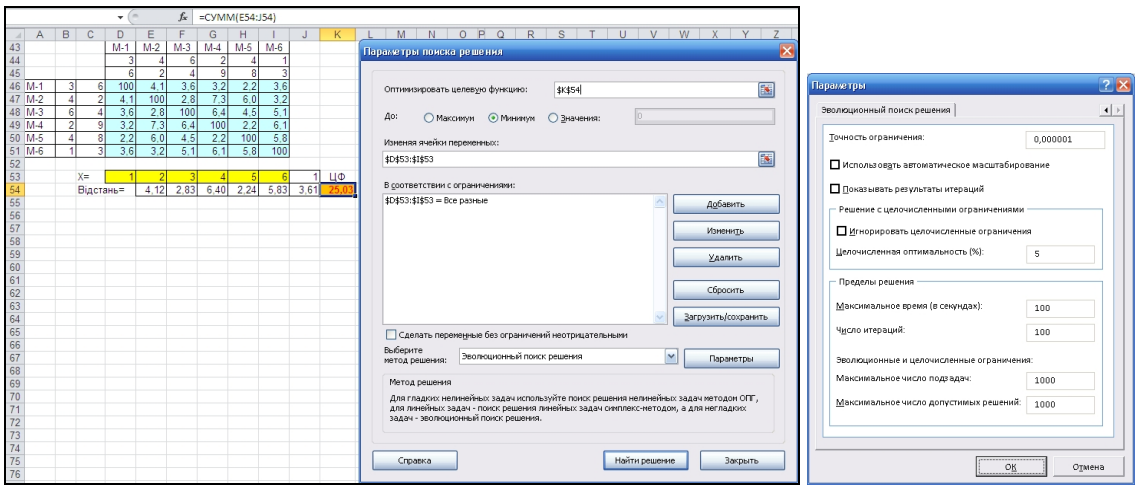
Починається процес обчислень — у рядку стану внизу вікна поступово виводиться повідомлення такого вигляду



де видно: нове значення ЦФ, номер підзадачі і попереднє значення ЦФ.

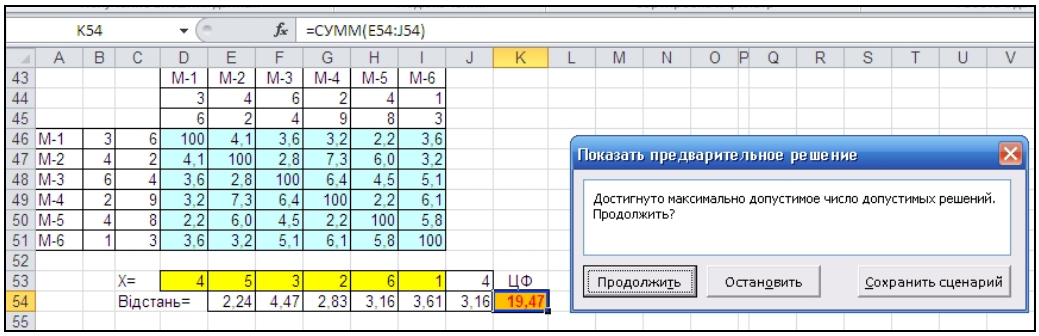
Процес припиняється, коли значення ЦФ не змінюється в межах заданої точності або вичерпані граничні значення заданих параметрів і пропонується альтернатива: *Продолжить*/*Остановить*.

Процес оптимізаційного моделювання завершується за командою *Остановить*.



Результат.

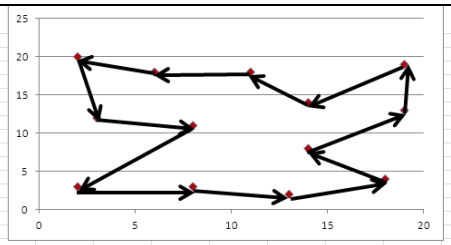
Знайдено оптимальний контур обходу $X = \{4, 5, 3, 2, 6, 1, 4\}$ із мінімальною довжиною 19,47. Результат співпав з усіма попередніми результатами, але отримано його значно швидше з мінімальними витратами часу та розміру документа:



Усі наступні приклади реалізовані за допомогою еволюційного методу, бо з урахуванням розмірів цих задач альтернативи цьому методу нема. Початковими даними є координати географічних об'єктів чи тестові дані, що отримані за допомогою генератора випадкових чисел.

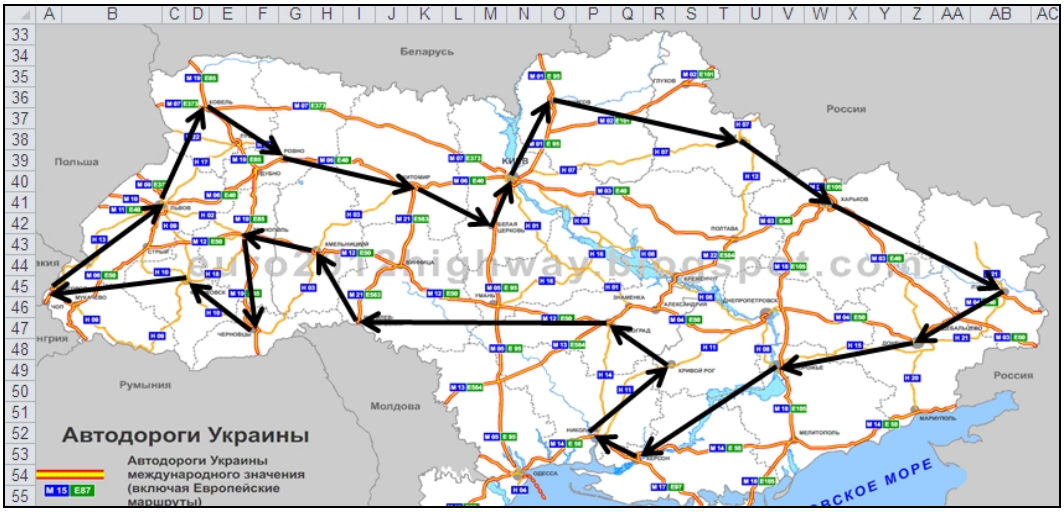
Приклад 4 ($n = 13$).

			М-1	М-2	М-3	М-4	М-5	М-6	М-7	М-8	М-9	М-10	М-11	М-12	М-13	
			2	3	6	8	8	14	14	11	19	19	18	2	13	
			3	12	18	11	3	8	14	18	19	13	4	20	2	
1	М-1	2	3	999,0	9,1	15,5	10,0	6,0	13,0	16,3	17,5	23,3	19,7	16,0	17,0	11,0	1
2	М-2	3	12	9,1	999,0	6,7	5,1	10,3	11,7	11,2	10,0	17,5	16,0	17,0	8,1	14,1	1
3	М-3	6	18	15,5	6,7	999,0	7,3	15,1	12,8	8,9	5,0	13,0	13,9	18,4	4,5	17,5	1
4	М-4	8	11	10,0	5,1	7,3	999,0	8,0	6,7	6,7	7,6	13,6	11,2	12,2	10,8	10,3	1
5	М-5	8	3	6,0	10,3	15,1	8,0	999,0	7,8	12,5	15,3	19,4	14,9	10,0	18,0	5,1	1
6	М-6	14	8	13,0	11,7	12,8	6,7	7,8	999,0	6,0	10,4	12,1	7,1	5,7	17,0	6,1	1
7	М-7	14	14	16,3	11,2	8,9	6,7	12,5	6,0	999,0	5,0	7,1	5,1	10,8	13,4	12,0	1
8	М-8	11	18	17,5	10,0	5,0	7,6	15,3	10,4	5,0	999,0	8,1	9,4	15,7	9,2	16,1	1
9	М-9	19	19	23,3	17,5	13,0	13,6	19,4	12,1	7,1	8,1	999,0	6,0	15,0	17,0	18,0	1
10	М-10	19	13	19,7	16,0	13,9	11,2	14,9	7,1	5,1	9,4	6,0	999,0	9,1	18,4	12,5	1
11	М-11	18	4	16,0	17,0	18,4	12,2	10,0	5,7	10,8	15,7	15,0	9,1	999,0	22,6	5,4	1
12	М-12	2	20	17,0	8,1	4,5	10,8	18,0	17,0	13,4	9,2	17,0	18,4	22,6	999,0	21,1	1
13	М-13	13	2	11,0	14,1	17,5	10,3	5,1	6,1	12,0	16,1	18,0	12,5	5,4	21,1	999,0	1
				1	1	1	1	1	1	1	1	1	1	1	1	1	
	План			9	7	8	3	12	2	4	1	5	13	11	6	10	5	ЦФ
	Відстань				7,07	5	5	4,47	8,06	5,1	10	6	5,1	5,39	5,66	7,07	6	79,92

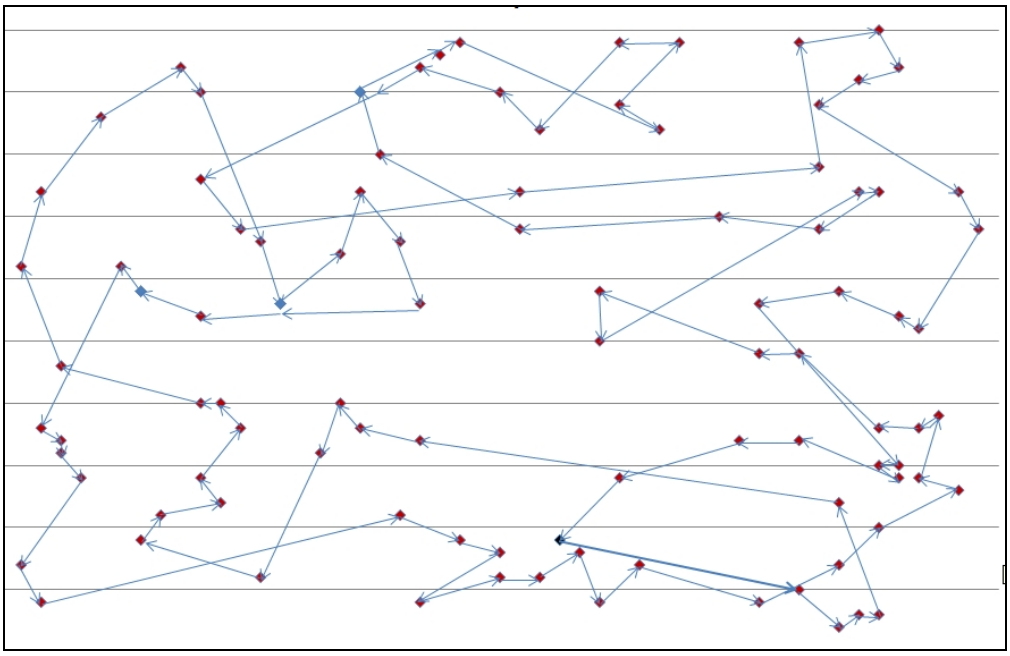


Приклад 5 ($n = 22$).

				Ужгород	Львів	Ів.-Фр.	Рівне	Чернівці	Ковель	Хмельницький	Мог.-Под.	Тернопіль	Житомир	Київ	Чернігів	Суми	Б.-Церква	Кіровоград	Миколаїв	Херсон	Запоріжжя	Донецьк	Луганськ	Харків	Кр.-Ріг	
				1	5	6	9	8	6	10	12	8	14	18	18	25	17	21	20	22	27	31	34	28	23	
				11	15	11	17	9	20	13	9	13	16	16	20	17	14	9	4	2	7	8	10	15	7	
1	Ужгород	1	11	999	5,7	5,0	10,0	7,3	10,3	9,2	11,2	7,3	13,9	17,7	19,2	24,7	16,3	20,1	20,2	22,8	26,3	30,1	33,0	27,3	22,4	
2	Львів	5	15	5,7	999	4,1	4,5	6,7	5,1	5,4	9,2	3,6	9,1	13,0	13,9	20,1	12,0	17,1	18,6	21,4	23,4	26,9	29,4	23,0	19,7	
3	Ів.-Фр.	6	11	5,0	4,1	999	6,7	2,8	9,0	4,5	6,3	2,8	9,4	13,0	15,0	19,9	11,4	15,1	15,7	18,4	21,4	25,2	28,0	22,4	17,5	
4	Рівне	9	17	10,0	4,5	6,7	999	8,1	4,2	4,1	8,5	4,1	5,1	9,1	9,5	16,0	8,5	14,4	17,0	19,8	20,6	23,8	26,0	19,1	17,2	
5	Чернівці	8	9	7,3	6,7	2,8	8,1	999	11,2	4,5	4,0	4,0	9,2	12,2	14,9	18,8	10,3	13,0	13,0	15,7	19,1	23,0	26,0	20,9	15,1	
6	Ковель	6	20	10,3	5,1	9,0	4,2	11,2	999	8,1	12,5	7,3	8,9	12,6	12,0	19,2	12,5	18,6	21,3	24,1	24,7	27,7	29,7	22,6	21,4	
7	Хмельницький	10	13	9,2	5,4	4,5	4,1	4,5	8,1	999	4,5	2,0	5,0	8,5	10,6	15,5	7,1	11,7	13,5	16,3	18,0	21,6	24,2	18,1	14,3	
8	Мог.-Под.	12	9	11,2	9,2	6,3	8,5	4,0	12,5	4,5	999	5,7	7,3	9,2	12,5	15,3	7,1	9,0	9,4	12,2	15,1	19,0	22,0	17,1	11,2	
9	Тернопіль	8	13	7,3	3,6	2,8	4,1	4,0	7,3	2,0	5,7	999	6,7	10,4	12,2	17,5	9,1	13,6	15,0	17,8	19,9	23,5	26,2	20,1	16,2	
10	Житомир	14	16	13,9	9,1	9,4	5,1	9,2	8,9	5,0	7,3	6,7	999	4,0	5,7	11,0	3,6	9,9	13,4	16,1	15,8	18,8	20,9	14,0	12,7	
11	Київ	18	16	17,7	13,0	13,0	9,1	12,2	12,6	8,5	9,2	10,4	4,0	999	4,0	7,1	2,2	7,6	12,2	14,6	12,7	15,3	17,1	10,0	10,3	
12	Чернігів	18	20	19,2	13,9	15,0	9,5	14,9	12,0	10,6	12,5	12,2	5,7	4,0	999	7,6	6,1	11,4	16,1	18,4	15,8	17,7	18,9	11,2	13,9	
13	Суми	25	17	24,7	20,1	19,9	16,0	18,8	19,2	15,5	15,3	17,5	11,0	7,1	7,6	999	8,5	8,9	13,9	15,3	10,2	10,8	11,4	3,6	10,2	
14	Б.-Церква	17	14	16,3	12,0	11,4	8,5	10,3	12,5	7,1	7,1	9,1	3,6	2,2	6,1	8,5	999	6,4	10,4	13,0	12,2	15,2	17,5	11,0	9,2	
15	Кіровоград	21	9	20,1	17,1	15,1	14,4	13,0	18,6	11,7	9,0	13,6	9,9	7,6	11,4	8,9	6,4	999	5,1	7,1	6,3	10,0	13,0	9,2	2,8	
16	Миколаїв	20	4	20,2	18,6	15,7	17,0	13,0	21,3	13,5	9,4	15,0	13,4	12,2	16,1	13,9	10,4	5,1	999	2,8	7,6	11,7	15,2	13,6	4,2	
17	Херсон	22	2	22,8	21,4	18,4	19,8	15,7	24,1	16,3	12,2	17,8	16,1	14,6	18,4	15,3	13,0	7,1	2,8	999	7,1	10,8	14,4	14,3	5,1	
18	Запоріжжя	27	7	26,3	23,4	21,4	20,6	19,1	24,7	18,0	15,1	19,9	15,8	12,7	15,8	10,2	12,2	6,3	7,6	7,1	999	4,1	7,6	8,1	4,0	
19	Донецьк	31	8	30,1	26,9	25,2	23,8	23,0	27,7	21,6	19,0	23,5	18,8	15,3	17,7	10,8	15,2	10,0	11,7	10,8	4,1	999	3,6	7,6	8,1	
20	Луганськ	34	10	33,0	29,4	28,0	26,0	26,0	29,7	24,2	22,0	26,2	20,9	17,1	18,9	11,4	17,5	13,0	15,2	14,4	7,6	3,6	999	7,8	11,4	
21	Харків	28	15	27,3	23,0	22,4	19,1	20,9	22,6	18,1	17,1	20,1	14,0	10,0	11,2	3,6	11,0	9,2	13,6	14,3	8,1	7,6	7,8	999	9,4	
22	Кр.-Ріг	23	7	22,4	19,7	17,5	17,2	15,1	21,4	14,3	11,2	16,2	12,7	10,3	13,9	10,2	9,2	2,8	4,2	5,1	4,0	8,1	11,4	9,4	999	
	План			3	1	2	6	4	10	14	11	12	13	21	20	19	18	17	16	22	15	8	7	9	5	3
	Відстань				5,0	5,7	5,1	4,2	5,1	3,6	2,2	4,0	7,6	3,6	7,8	3,6	4,1	7,1	2,8	4,2	2,8	9,0	4,5	2,0	4,0	2,8	100,97



Приклад 6 (n = 100).

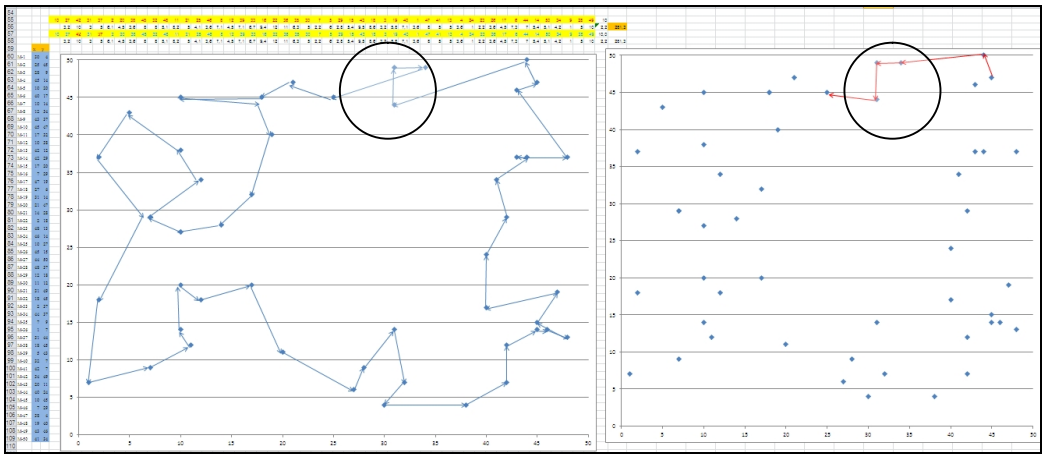


Корегування результату.

Доведено [1], що найкоротший замкнений контур, що зв'язує n точок на площині, є замкненою ламаною зі сторонами, що не перетинаються. Наближений розв'язок задачі великого розміру може утворити перетини у вигляді петель, тож їхня заміна ламаними — шлях до покращення шуканого результату.

Приклад 7.

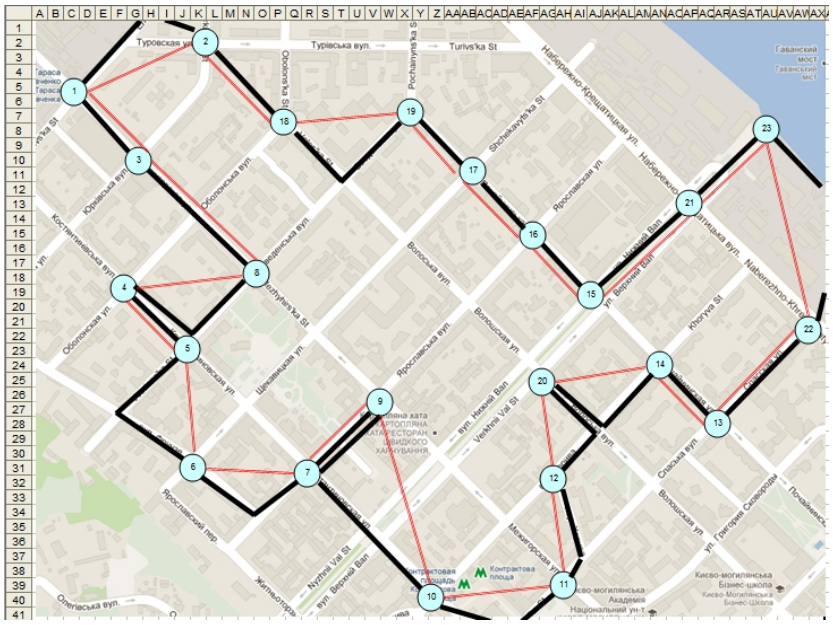
За допомогою еволюційного методу розв'язана задача комівояжера розміром 50×50, значення ЦФ склало величину 289,4, побудована схема контуру (зліва).



На діаграмі визначена петля, яка вручну замінена ламаною, що привело до зменшеного значення ЦФ величиною 281,3. Відповідний фрагмент оновленої

комбінації номерів уведений у модель, і повторний її запуск підтвердив оновлений результат. Таким чином, для задач великого розміру з'являється можливість дещо покращити отриманий наближений результат шляхом корекції певних фрагментів діаграми у вигляді перетинань.

Приклад 8. Манхетенська метрика.
Задача: обійти 23 пункти на карті Києва, рухаючись вулицями.
Результат.



1. *Басакер Р.* Конечные графы и сети / Р. Басакер, Т. Саати. — М.: Наука, 1974. — 366 с.

2. *Васильев В.В.* Гибридные модели задач оптимизации / В.В. Васильев, А.Г. Додонов. — К.: Наук. думка, 1974. — 215 с.

3. *Кристофидес Н.* Теория графов: Алгоритмический подход / Н. Кристофидес / Пер. с англ. — М.: Мир, 1978. — 432 с.

4. *Fylstra D.* Design and Use of the Microsoft Excel Solver / D. Fylstra, L. Lasdon. — INTERFACES. — 1998. — Vol. 28, N 5. — P. 29–55.

5. *Powell S.* The Art of Modeling with Spreadsheets: Management Science, Spreadsheet Engineering and Modeling Craft / S. Powell, K. Baker. — John Wiley & Sons, 2004. — 400 p.

6. *Кузьмичов А.І.* Математичне програмування в Excel / А.І. Кузьмичов, М.Г. Медведєв. — К.: Вид-во Європ. ун-ту, 2005. — 320 с.

7. *Кузьмичов А.І.* Таблична реалізація генетичного алгоритму пошуку рішень для нелінійних задач оптимізації / А.І. Кузьмичов // Наук. вісник АМУ. Серія «Техніка». Автоматизація та комп'ютерно-інтегровані технології управління. — К.: АМУ, 2008. — С. 125–135.

8. *Кузьмичов А.І.* Таблична реалізація генетичного алгоритму пошуку рішень для «важких» задач оптимізації / А.І. Кузьмичов // Тр. межд. конф. «Моделирование-2008», ИПМЭ НАНУ. — 2008. — Т. 1. — С. 94–108.

Надійшла до редакції 19.07.2011