УДК 62-50

Бідюк П.І., Литинська А.Ю.

МЕТОДИ ЛІНІЙНОГО ПРОГРАМУВАННЯ В ЗАДАЧАХ ФОРМУВАННЯ ПОРТФЕЛЯ ФІНАНСОВИХ ІНСТРУМЕНТІВ

Формування оптимального портфеля цінних паперів пропонується виконати за модифікованим симплекс-методом, який використовується для розв'язання задачі лінійного програмування. Переваги такого методу полягають у зменшенні числа обчислювальних операцій, витрат оперативної пам'яті, підвищенні швидкості збіжності і відсутності ефекту зациклення. Отримані результати порівнюються із звичайним симплекс-методом.

Ключові слова: *модифікований симплекс-метод, портфель цінних паперів, індекс хедж-фонда.*

Формирование оптимального портфеля ценных бумаг предлагается осуществить с помощью модифицированного симплекс-метода, который используется для решения задачи линейного программирования. Преимущества этого подхода заключаются в уменьшении количества вычислительных операций, объема требуемой памяти, ускорении сходимости и отсутствии эффекта зацикливания. Полученные результаты сравниваются с обычным симплекс-методом.

Ключевые слова: *модифицированный симплекс-метод, портфель ценных бумаг, индекс хедж-фонда.*

To create optimal financial portfolio it is proposed to use the modified simplex method directed to solution of linear programming problem. The advantages of the approach are as follows: decreasing of computing operations number, memory requirements, improvement of convergence characteristics and elimination of possible unexpected loops. The results achieved are compared to commonly used simplex algorithm.

Key words: *modified simplex method, financial portfolio, index hedsch fonds.*

При розв'язанні багатьох задач із різних прикладних областей широке застосування знаходять методи лінійної оптимізації, також важко переоцінити їх роль в області фінансової аналітики. Зокрема, до задачі лінійного програмування зводиться більшість алгоритмів формування портфеля цінних паперів.

Як математична дисципліна, лінійна оптимізація веде відлік з основної роботи Л.В. Канторовича [1], кожний із розділів якої присвячено моделюванню конкретної економічної задачі. У цій же роботі представлено метод розв'язних множників – прообраз розробленого в 1947 році Дж. Данцигом симплекс-методу [2]. Завдяки простоті реалізації і високим швидкісним характеристикам модифікації симплекс-методу стали найпоширенішим способом розв'язання задач лінійного програмування.

Однак, симплекс-метод – далеко не єдиний спосіб розв'язання задач лінійного програмування. Зокрема, в 60-70-х роках зародився альтернативний напрямок – методи внутрішніх точок, перший з яких був опублікований в 1967 році І.І. Дікіним. Їх назва пов'язана з тим, що на відміну від симплекс-методу, який перебирає кутові точки багатогранника

припустимих розв'язків, обчислювальний процес у методах внутрішніх точок відбувається у відносній внутрішній області припустимої множини. Схематично траєкторії обох класів методів зображено на рис. 1.

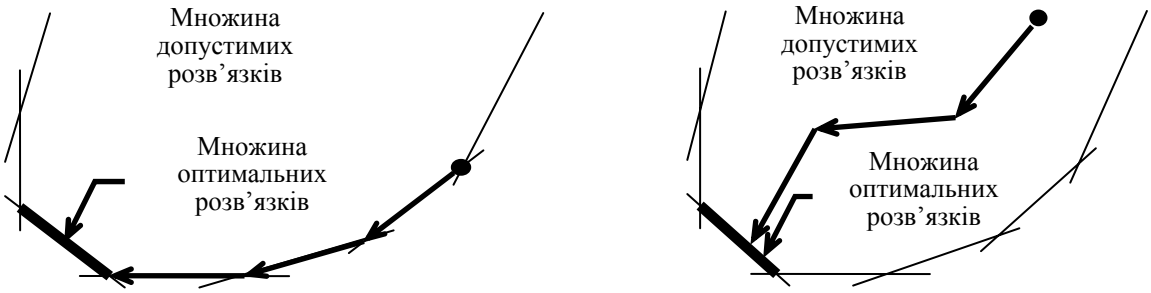


Рис. 1. Траєкторії симплекс-методу (ліворуч) та методу внутрішніх точок (праворуч)

Дослідження, які описуються в даній статті, виконані в рамках створення програмного продукту, призначеного для використання фінансовими аналітиками. В основу проекту закладено ідеї оптимізаційного підходу, який, у свою чергу, зобов'язує розробника використовувати передові технології чисельних методів. У даному випадку це методи лінійного програмування. Отже, необхідно виявити, який з двох підходів може задовольнити вимоги стосовно швидкодії та якості розв'язку поставленої задачі.

ПОСТАНОВКА ЗАДАЧІ

Перш ніж сформулювати оптимізаційні задачі, для розв'язання яких здійснювався пошук найбільш придатних і обґрунтованих методів лінійного програмування, необхідно ввести деякі поняття. Позначимо через x вектор, що описує портфель фінансових інструментів, де x_i характеризує позицію по інструменту i. Припускається, що випадковий вектор y визначається ймовірнісною мірою P на Y (Борелівська міра), яка не залежить від x і визначає прибутковість кожного інструменту портфеля. Для кожного x введемо через $\psi(x,\cdot)$ на $\Re$ – результуючу функцію розподілу втрат $z = f(x,y)$, тобто:

$$\psi(x,\xi) = P\{y \mid f(x,y) \le \xi\}. \tag{1.1}$$

Тепер можна дати означення для VaR.

***Означення VaR*:** $\alpha - VaR$ втрати, які відповідають розв'язку x, – це величина, що дорівнює

$$\xi_\alpha(x) = \min\{\xi \mid \psi(x,\xi) \ge \alpha\}. \tag{1.2}$$

Коли $\psi(x,\cdot)$ неперервна і строго зростаюча, $\xi_\alpha(x)$ єдина ξ, яка задовольняє $\psi(x,\xi) = \alpha$. В іншому випадку рівняння може не мати розв'язків або мати їх безліч.

***Означення CVaR*:** $\alpha - CVaR$ – втрати, які відповідають розв'язку x; це величина, що дорівнює $\phi_\alpha(x) =$ математичному сподіванню α-хвоста розподілу $z = f(x,y)$, де описаний розподіл визначається так:

$$\psi_\alpha(x,\xi) = \begin{cases} 0, & \text{если } \xi_\alpha < \xi_\alpha(x), \\ \left[\psi(x,\xi) - \alpha\right]/\left[1-\alpha\right], & \text{если } \xi_\alpha \ge \xi_\alpha(x). \end{cases} \tag{1.3}$$

Цей підхід запропонований Урясевим та Рокафеларом [3].

У даній роботі як міра ризику використовується *Conditional Drawdown – at-Risk* (CDaR), яка запропонована в роботі [2]. Вона визначається аналогічно до CVaR, але має специфічну функцію втрат:

$$f(x,t) = \max_{1 \le \tau \le t}\{w(x,\tau)\} - w(x,t), \tag{1.4}$$

де $w(x,t)$ – вартість портфеля на момент часу *t*.

***Означення CDaR*:** $\alpha - CDaR$ – втрати, які відповідають розв'язку *x*; це величина, що дорівнює:

$$\Delta_\alpha(x) = \frac{1}{(1-\alpha)T}\int_\Omega f(x,t)dt, \quad \Omega = \{t \in [0,T] : f(x,t) \geq \beta(x,\alpha)\}, \tag{1.5}$$

де $\beta(x,\alpha)$ – це поріг втрат, який перевищує тільки $(1-\alpha)\cdot 100\ \%$ сценаріїв.

З урахуванням цих мір ризику після проведення дискретизації можемо сформулювати такі оптимізаційні задачі.

1) Пошук оптимального портфеля цінних паперів шляхом мінімізації CVaR:

$$\min_{(x,\xi)\in X\times R}\left[\xi + \frac{1}{(1-\alpha)q}\sum_{i=1}^{q}\left[f(x,y_i)-\xi\right]^+\right], \tag{1.6}$$

або у зручнішому вигляді:

$$\min_{(x,\xi)\in X\times R}\left[\xi + \frac{1}{(1-\alpha)q}\sum_{i=1}^{q} z_i\right], \tag{1.7}$$

$$z_i \geq \left[f(x,y_i)-\xi\right],$$

$$z_i \geq 0, \quad i = \overline{1..q}.$$

2) Пошук оптимального портфеля цінних паперів шляхом мінімізації CDaR:

$$\min_{(x,\xi)\in X\times R}\left[\beta + \frac{1}{(1-\alpha)q}\sum_{i=1}^{q}\left[\max_{1\leq\tau\leq i}\{w(x,\tau)\} - w(x,i) - \beta\right]^+\right], \tag{1.8}$$

або

$$\min_{(x,\xi)\in X\times R}\left[\beta + \frac{1}{(1-\alpha)q}\sum_{i=1}^{q} z_i\right], \tag{1.9}$$

$$z_i \geq t_i - w(x,i) - \beta, \quad i = \overline{1..q},$$

$$t_k \geq w(x,k), \quad k = \overline{1..q},$$

$$t_k \geq t_{k-1}, \quad k = \overline{1..q},$$

$$t_0 = 0,$$

$$z_i \geq 0, \quad i = \overline{1..q}.$$

Афінно-масштабуючі алгоритми. Джерела алгоритмів внутрішніх точок виходять із запропонованої в 1965 році Л.В. Канторовичем методики оцінювання множників Лагранжа – обмежень задачі при неоптимальному плані за методом найменших квадратів. Ця методика докладно описана в [5].

В одному з її варіантів для припустимого, але неоптимального розв'язку $\mathbf{x^k} \in ri\mathbf{X}$ задачі

$$\mathbf{c^T x} \to \min_{\mathbf{x}\in\mathbf{X}}, \mathbf{X} = \left\{\mathbf{x} \in \mathbf{R^n} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\}, \tag{2.1}$$

вектор оцінок визначається за формулою:

$$\mathbf{u^k} = \arg\min_{\mathbf{u}\in\mathbf{R^m}} \sum_{j=1}^{n} d_j^k \left(g_j(\mathbf{u})\right)^2, \tag{2.2}$$

де

$$d_j^k = \left(x_j^k\right)^2, j = 1,...,n. \tag{2.3}$$

На основі цього правила І.І. Дікіним [5] у 1967 році був розроблений перший алгоритм внутрішніх точок для задачі лінійного програмування, у якому розв'язок покращується ітеративно за правилом:

$$\mathbf{x}^{\mathbf{k+1}} = \mathbf{x}^{\mathbf{k}} + \lambda_k \Delta\mathbf{x}^{\mathbf{k}},\ k = 1,2,\ldots, \tag{2.4}$$

де $\Delta\mathbf{x}^{\mathbf{k}}$ – напрям корегування, який обчислюється за формулою:

$$\Delta x_j^k = -d_j^k g_j\left(\mathbf{u}^{\mathbf{k}}\right), j = 1,\ldots,n, \tag{2.5}$$

а λ_k – крок корегування, який визначається рівністю:

$$\lambda_k = 1 \Big/ \sqrt{\sum_{j=1}^{n} d_j^k \left(g_j\left(\mathbf{u}^{\mathbf{k}}\right)\right)^2}\ . \tag{2.6}$$

Геометрично вектор $\mathbf{x}^{\mathbf{k+1}}$ у даному алгоритмі визначається як точка мінімуму цільової функції задачі (2.1) на вписаному в $\mathbf{X}$ еліпсоїді із центром в $\mathbf{x}^{\mathbf{k}}$.

В.І. Зоркальцев запропонував у роботах [6, 10] ще один варіант алгоритму (2.2)-(2.6), який ґрунтується на ідеї руху вздовж напряму корегування $\Delta\mathbf{x}^{\mathbf{k}}$ не до границі еліпсоїда, а на частину шляху, рівну γ, – від точки $\mathbf{x}^{\mathbf{k}}$ до границі додатного ортанта. У цьому випадку крок корегування обчислюється не у відповідності до (2.6), а наступним чином:

$$\lambda_k = \gamma \min_{j:\Delta x_j^k < 0} \left(-x_j^k / \Delta x_j^k\right),\ \gamma \in (0;2/3). \tag{2.7}$$

Обмеження, що накладають на γ, є необхідними для обґрунтування алгоритму у випадку відсутності припущення стосовно невиродженості задачі, хоча при використанні алгоритму на практиці (де вироджені задачі зустрічаються досить рідко) у програмно-обчислювальних комплексах рекомендуються до застосування такі величини: $\gamma = 0.95$, $\gamma = 0.99$, а показники γ^k змінюються в залежності від ітерації і асимптотично наближуються до одиниці.

Введення в допустиму область. Для прямих афінно-масштабуючих (АМ) алгоритмів необхідно мати стартову точку $\mathbf{x} \in \mathbf{X}$, для якої обмеження-нерівності виконуються в строгій формі. Оскільки така точка не завжди апріорно відома, то за допомогою незначних модифікацій алгоритму можна сумістити процеси оптимізації і введення в допустиму область. Обчислювальний процес при цьому починається з будь-якого вектора $\mathbf{x}^{\mathbf{1}} > \mathbf{0}$. На кожній ітерації обчислюється вектор нев'язок балансових обмежень-рівностей $\mathbf{r}^{\mathbf{k}} = \mathbf{b} - \mathbf{A}\mathbf{x}^{\mathbf{k}}$. Для визначення напряму корегування замість задачі (2.1) розв'язується така:

$$\mathbf{c}^{\mathbf{T}}\Delta\mathbf{x} + \frac{1}{2}\sum_{j=1}^{n} \frac{\Delta x_j^2}{d_j^k} \to \min_{\Delta\mathbf{x} \in \mathbf{R}^{\mathbf{n}}},\ \mathbf{A}\Delta\mathbf{x} = \mathbf{r}^{\mathbf{k}}. \tag{2.8}$$

В усьому іншому алгоритм залишається незмінним. Нев'язки обмежень-рівностей ітераційно скорочуються за правилом: $\mathbf{r}^{\mathbf{k+1}} = (1 - \lambda_k)\mathbf{r}^{\mathbf{k}}$. Тому при $\mathbf{x} \notin \mathbf{X}$ крок корегування λ_k повинен обмежуватися зверху одиницею, а якщо $\mathbf{r}^{\mathbf{k}} \neq \mathbf{0}$, то потрібно повторно обчислити $\lambda_k := \min\{1, \lambda_k\}$.

Двоїсті алгоритми. У процесі розв'язання задачі на кожній ітерації обчислюються двоїсті оцінки $\mathbf{u}^{\mathbf{k}}$, які, хоча й немонотонно, сходяться до оптимального розв'язку:

$$\mathbf{b}^{\mathbf{T}}\mathbf{u} \to \max_{\mathbf{u} \in \mathbf{U}},\ \mathbf{U} = \left\{\mathbf{u} \in \mathbf{R}^{\mathbf{m}} : \mathbf{g}(\mathbf{u}) \equiv \mathbf{c} - \mathbf{A}^{\mathbf{T}}\mathbf{u} \geq \mathbf{0}\right\}. \tag{2.7}$$

Існують теоретичні результати, які показують, що двоїсті оцінки збігаються швидше ніж змінні прямої задачі. Обчислювальний процес у двоїстих алгоритмах починається з довільних векторів $\mathbf{u}^{\mathbf{1}}$ й $\mathbf{g}^{\mathbf{1}} > \mathbf{0}$. На кожній ітерації обчислюється вектор нев'язок обмежень-рівностей $\mathbf{r}^{\mathbf{k}} = \mathbf{c} - \mathbf{A}^{\mathbf{T}}\mathbf{u}^{\mathbf{k}} - \mathbf{g}^{\mathbf{k}}$. Напрямки корегування змінних $\Delta\mathbf{u}^{\mathbf{k}}$ і $\Delta\mathbf{g}^{\mathbf{k}}$ знаходять шляхом розв'язання такої задачі:

$$-\mathbf{b}^{\mathrm{T}}\Delta\mathbf{u}+\frac{1}{2}\sum_{j=1}^{n}\frac{\left(\Delta g_j\right)^2}{d_j^k}\to\min_{\Delta\mathbf{u}\in\mathbf{R}^{\mathbf{m}},\Delta\mathbf{g}\in\mathbf{R}^{\mathbf{n}}},\Delta\mathbf{g}+\mathbf{A}^{\mathrm{T}}\Delta\mathbf{u}=\mathbf{r}^{\mathbf{k}},\tag{2.8}$$

де d_j^k – вагові коефіцієнти, які можна обчислити за виразом: $d_j^k=\min\left\{\left(g_j^k\right)^p,N\right\},j=1,...,n$, $p\geq 1$.

Крок корегування обчислюється аналогічно (2.7):

$$\lambda_k=\gamma\min_{j:\Delta g_j^k<0}\left(-g_j^k/\Delta g_j^k\right),\ \gamma\in\left(0;2/\left(p+1\right)\right).$$

Модифікований симплекс-метод. Основна ідея симплекс-методу (СМ), одного з найбільш поширених алгоритмів розв'язку задач лінійного програмування, полягає у покращенні цільової функції в процесі руху від однієї крайньої точки допустимої множини до іншої, сусідньої.

СМ дозволяє знайти оптимальний стан або дійти до висновку, що задача не має розв'язку шляхом визначення тільки допустимих базових розв'язків (БР). Для задачі, представленої матричному вигляді (2.1), алгоритм методу є таким [7].

1) Визначення поточного БР:

$$X=B^{-1}\cdot b,\tag{2.9}$$

де X – поточний розв'язок; B^{-1} – обернена матриця.

2) Знаходження змінної, яка має увійти до БР. Для всіх небазисних векторів визначаємо симплекс-різницю:

$$\widetilde{c}=c_V\ -c_B\cdot B^{-1}\cdot V,\quad j:\left\{c_j=\min_i(c_i)\right\},\tag{2.10}$$

де c_V – вектор коефіцієнтів цільової функції, для $x_i\notin$ БР; c_B – вектор коефіцієнтів цільової функції; $x_i\in$БР; V – матриця, яка складається з векторів А, що не увійшли до БР.

Якщо $\widetilde{c}_i\geq 0,i=\overline{1,N}$, то поточний розв'язок оптимальний, інакше йдемо на наступний крок.

3) Визначаємо змінну, яка має залишити БР:

$$w_j=B^{-1}\cdot A_j,\ \forall\ j\notin БР,\quad i:\left\{\frac{d_i}{w_i}=\min_t(\frac{d_t}{w_t}),w_t>0\right\},\tag{2.11}$$

якщо $\forall t:w_t\leq 0$, то розв'язок необмежений.

4) Повторно обчислюємо матрицю B.

Послідовні базиси на ітераціях СМ відрізняються на кожному кроці лише одним вектором, який вводиться в БР. Саме на цьому грунтується мультиплікативне представлення матриці B:

$$B=T_1\cdot T_2\cdots T_l\ \ B^{-1}=T_l^{-1}\cdot T_{l-1}{}^{-1}\cdots T_1^{-1},\tag{2.12}$$

де l – кількість ітерацій, а T_k отримують з одиничної матриці $m\times m$; таким чином маємо [8]:

$$T_k = \begin{bmatrix} 1 & & t_{1k} & & \\ & 1 & \cdot & & \\ & & \cdot & \cdot & \\ & & \cdot & & \\ & & t_{kk} & & \\ & & \cdot & \cdot & \\ & & \cdot & & \cdot \\ & & \cdot & & \cdot \\ & & t_{mk} & & 1 \end{bmatrix}, \; t_k = \frac{1}{(B^{-1} \cdot A_j)_k} \begin{bmatrix} (B^{-1} \cdot A_j)_1 \\ \cdot \\ \cdot \\ (B^{-1} \cdot A_j)_{k-1} \\ 1 \\ (B^{-1} \cdot A_j)_{k+1} \\ \cdot \\ \cdot \\ (B^{-1} \cdot A_j)_m \end{bmatrix} \qquad (2.13)$$

де k – індекс змінної, що вводится до БР; j – індекс змінної, що виводиться з алгоритму. Таким чином, немає необхідності перераховувати на кожній ітерації матрицю B та зберігати її в пам'яті. Достатньо мати перехідні вектори t_k, з яких за формулою (2.12) завжди можна отримати B. Цей алгоритм також дозволяє уникнути безпосередньої операції обернення матриці та пов'язаних із цим додаткових арифметичних операцій, що значно підвищує швидкодію алгоритму у порівнянні з оригінальним.

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

Аналіз доцільності застосування описаних вище методів для розв'язку задач (1.7) і (1.9) виконано в системі MathCad, а програмний продукт реалізовано у системі C++ Builder.

Як фінансові інструменти використано індекси хедж-фондів, надані компанією CSFB/TREMONT. Маючи підмножину з приблизно 650 фондів, вона надає інвесторам інформацію з 10-ти основних типів індексів, які були обрані для виконання експерименту (Convertible Arbitrage**,** Emerging markets, Event driven, Distressed securities, Risk arbitrage, Fixed Income Arbitrage, Global Macro, Long/Short Equity, Managed Futures, Multi Strategy). Дані CSFB/TREMONT відображають величину зміни прибутковості середньозваженого портфеля хедж-фондів за місяць. Для роботи обрані показники за період з 2000 до 2006 року.

Потрібно також брати до уваги деякі особливості матриць обмежень оптимізаційних задач (1.7) та (1.9). По-перше, їх розмірність безпосередньо залежить від величини псевдовипадкового ряду, що генерується. А в задачах цього типу він може налічувати десятки тисяч значень. По-друге, вони мають досить низький відсоток ненульових елементів, який не перевищує 3 %. Отже, маємо справу з розрідженою матрицею великої розмірності.

У табл. 1 і 2 представлено результати експерименту, які показують залежність кількості ітерацій методу від розміру ряду, що генерується.

Таблиця 1

Задача (1.7)

Розмір вибірки	Двоїстий АМ алгоритм	Модифікований СМ
10	92	4
50	438	7
100	501	17
150	629	27
200	713	31

Таблиця 2

Задача (1.9)

Розмір вибірки	Двоїстий АМ алгоритм	Модифікований СМ
10	74	33
50	236	26
100	377	78
150	533	86
200	921	131

Отже, з'ясувалось, що двоїстий АМ алгоритм зовсім не можна застосовувати для розв'язання задач такого виду. Ці результати зумовлені самою структурою матриці обмежень та її поганою обумовленістю. У векторі розв'язку також переважають нульові елементи, у той

час як шукані змінні відрізняються на декілька порядків. Саме це не дозволяє АМ алгоритму працювати ефективно. В процесі дослідження також спостерігались випадки, коли метод наближував оптимальний розв'язок із значною похибкою, що є неприпустимим у задачах такого цільового призначення.

СМ, навпаки, виявився стійким, достатньо швидкодіючим та економічним з точки зору інформації, яку потрібно зберігати в пам'яті комп'ютера під час роботи алгоритму. Обчислювальна складність модифікованого СМ нижча за АМ алгоритм. А відсутність операції обернення матриці обмежень робить його стійким навіть у випадку її поганої обумовленості.

СМ, як один із найпоширеніших оптимізаційних алгоритмів, має багато варіацій, розрахованих на різні типи задач. Наприклад, дослідження виконане в [9], показує, що вигляд критерію (2.10), згідно з яким вибирається змінна для введення в БР, може істотно впливати на швидкість збіжності методу. Саме тому наступним кроком дослідження став порівняльний аналіз декількох правил.

Отже, маємо вектор оцінок $\widetilde{c} = c_V - c_B \cdot B^{-1} \cdot V$. Нехай $R_r = \left\{ j \in R \,|\, \widetilde{c}_j < 0 \right\}$

Стратегія 1. $j : \left\{ \widetilde{c}_j = \min\limits_{i \in R_r} (\widetilde{c}_i) \right\}$

Стратегія 2. $c_j = \dfrac{\widetilde{c}_j}{\sqrt{1 + \sum\limits_{i=1}^{M} a_{i,j} \cdot (h1_i)^2}}, \quad j : \left\{ c_j = \max\limits_{i \in R_r} (c_i) \right\},$

де $h1_i$ – кількість ненульових елементів в *i*-му рядку матриці *A*.

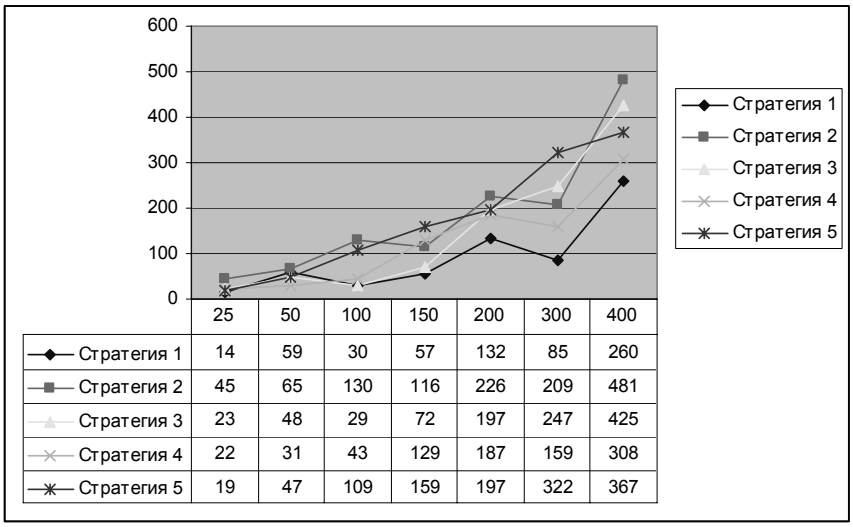
Стратегія 3. $j : \left\{ c_j = \min\limits_{i \in R_r} (random(\widetilde{c}_i)) \right\}.$

Стратегія 4. $c_j = \dfrac{\widetilde{c}_j}{1 + h2_j^{\,2}}, \quad j : \left\{ c_j = \max\limits_{i \in R_r} (c_i) \right\},$

де $h2_i$ – кількість ненульових елементів в *i*-му стовпчику матриці *A*.

Стратегія 5. $c_j = \dfrac{\widetilde{c}_j}{\sqrt{1 + \sum\limits_{i=1}^{M} a_{i,j}^{\,2}}}, \quad j : \left\{ c_j = \max\limits_{i \in R_r} (c_i) \right\},$

Результати, дослідження представлені на рис. 2.



	25	50	100	150	200	300	400
Стратегия 1	14	59	30	57	132	85	260
Стратегия 2	45	65	130	116	226	209	481
Стратегия 3	23	48	29	72	197	247	425
Стратегия 4	22	31	43	129	187	159	308
Стратегия 5	19	47	109	159	197	322	367

Рис. 2. Залежність кількості ітерацій СМ від типу стратегії та довжини ряду, що генерується

Як видно, найбільш результативними виявились стратегії 1 та 4. Саме вони були реалізовані в остаточній версії програмного продукту (рис. 3).

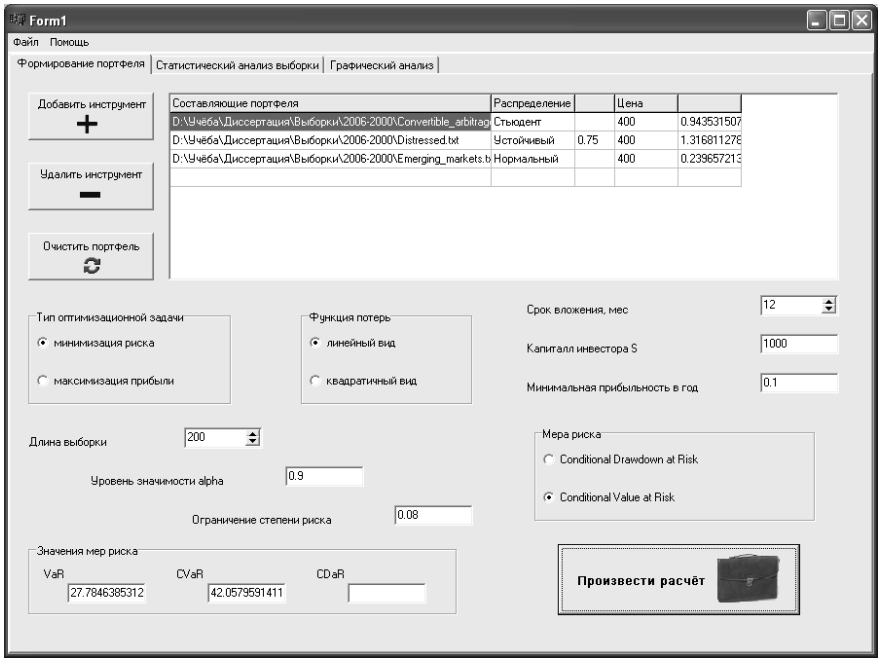


Рис. 3. Інтерфейс аналітичної системи, розробленої на основі задач (1.7) та (1.9)

ВИСНОВКИ

Ґрунтуючись на даних експерименту, можна зробити такі висновки. Алгоритми, що ґрунтуються на методах внутрішніх точок, не придатні для розв'язання задач вигляду (1.7) і (1.9), що пояснюється поганою обумовленістю матриць обмежень і її великою розмірністю. Функціонування симплекс-методу виявилася значно ефективнішим. Швидкість збіжності та стійкість алгоритму в процесі розв'язання задач (1.7) і (1.9) істотно перевершували показники алгоритмів внутрішніх точок. Також не було виявлено ефекту зациклення. Ще однією перевагою модифікованого СМ є відсутність операції знаходження оберненої матриці обмежень, що дозволяє знизити вплив її поганої обумовленості. Завдяки мультиплікативному представленню оберненої матриці значно скорочується обсяг інформації, необхідний для зберігання в пам'яті комп'ютера кожної ітерації. Використання різних стратегій вибору направляючого стовпця дозволяє збільшити швидкість збіжності алгоритму СМ.

Результатом виконаного дослідження також стало створення комп'ютерної інформаційно-аналітичної системи, призначеної для формування портфеля цінних паперів, зокрема портфеля хедж-фондів. У подальших дослідженнях планується застосувати запропонований метод формування фінансових інструментів до розв'язання практичних задач інвестування на різних рівнях інвестиційної діяльності.

ЛІТЕРАТУРА:

1. Канторович Л.В. Математические методы организации и планирования производства. – Л.: ЛГУ, 1939. – 68 с.
2. Данциг Дж. Линейное программирование, его применения и обобщения. –М.: Прогресс, 1966. – 600 с.
3. Rockafellar R. T., Uryasev S. Optimization of conditional value-at-risk // Journal of Risk, v. 2, pp. 21-41, www.ise.ufl.edu/uryasev/cvar.pdf.
4. Andersen E. D., Gondzio J., Meszaros C. Implementation of interior point methods for large scale linear programming / Technical report, 1996. – 45 p.
5. Дикин И.И., Зоркальцев В. И. Итеративное решение задач математического программирования (алгоритмы метода внутренних точек). – Новосибирск: Наука, 1980. – 144 с.

6. Зоркальцев В.И. Итеративный алгоритм решения задачи линейного программирования. В кн. Алгоритмы и программы решения задач линейной алгебры и математического программирования. – Иркутск: СЭИ СО АН СССР, 1978. – С. 77-89.
7. Коротков М., Гаврилов М., Основы линейного программирования. – М.: Высшая школа, 2003. – С. 25-28.
8. Tomlin J.A. Maintaining sparse inverse in the simplex method // IBM J. of Research and Development 1972, v. 16, № 4, pp. 415-423.
9. Djannaty F., Rostamy B. A computational approach to pivot selection in the LP relaxation of set problems // Journal of Applied Mathematics and Decision Sciences. – 2006. – № 1. – Pp. 1-12.
10. Зоркальцев В.И. Метод относительно внутренних точек. Сыктывкар: Коми филиал АН СССР, 1986. – 18 с.
11. Chekhlov A., Uryasev S., Zabarankin M., Portfolio optimization with drawdown constraints. Asset and Liability Management Tools /B. Scherer, ed. – London: Risk Books, 2000, pp. 263-278.

Рецензенти: д.т.н., проф. Коваленко І.І.,
д.т.н., проф. Казарєзов А.Я.

© Бідюк П.І., Литинська А.Ю., 2009 *Стаття надійшла до редакції 07.07.09*