

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА теории вероятностей

КУРСОВАЯ РАБОТА
специалиста
**Минимизация условной суммы под риском для
финансового портфеля**

Выполнила студентка 509 группы
Токаева Александра Александровна

подпись студента

Научный руководитель:

профессор, д.ф.-м.н.

Фалин Геннадий Иванович

подпись научного руководителя

Москва 2022

Оглавление

| | |
|---|----|
| 1. Введение | 3 |
| 2. Необходимые сведения | 4 |
| 3. Описание подхода | 4 |
| 3.1. Альтернативное доказательство Теоремы 2 | 7 |
| 4. Применение к оптимизации портфеля | 8 |
| 5. Результаты работы [1] для рассматриваемой задачи оптимизации портфеля | 11 |
| 6. Мои результаты в задаче оптимизации портфеля | 13 |
| 6.1. Реализация симплекс-метода glpk для задачи P1+P2 | 13 |
| 6.2. Реализация методов множителей Лагранжа в задаче P3 | 18 |
| 7. Применение к хеджированию | 21 |
| 8. Выводы | 25 |
| 9. Приложение | 25 |
| 10. Список литературы | 25 |

1. Введение

Задача выбора оптимального портфеля среди всех допустимых портфелей, обладающих заданными свойствами, является одной из основополагающих задач, встающих перед любыми участниками финансовых рынков. Именно поэтому данная задача уже много лет подвергается тщательному исследованию. Современная портфельная теория была предложена в 1952 году Гарри Марковитцем, и с тех пор было изобретено множество различных подходов к задаче оптимизации портфеля. Наиболее часто требуется добиться максимизации ожидаемого дохода или минимизации финансовых рисков.

Цель данной работы — исследовать новый подход к оптимизации портфеля финансовых инструментов для уменьшения риска больших потерь. Этот подход основан на минимизации $CVaR$, в то время как традиционные методы минимизируют VaR . Стоит отметить, что портфели с маленьким $CVaR$ обязательно имеют и маленькое VaR , то есть хорошие с точки зрения нового метода портфели будут хорошими и с точки зрения традиционных подходов. Центральным достижением нового подхода является техника, которая одновременно минимизирует $CVaR$ и находит VaR . Эта техника подойдет любой инвестиционной компании, брокерской фирме, фонду, а также любому бизнесу, который сталкивается с необходимостью оценивать риски. Кроме того, предложенную технику можно объединить с аналитическими или сценарными методами оптимизации портфелей с большим числом инструментов, причем в этих случаях вычисления сводятся к задачам линейного или негладкого программирования, для которых существуют определенные алгоритмы нахождения решения. Наконец, отметим, что данная техника подходит для оптимизации перцентилей в любой области, не только финансовой, что предоставляет широкие возможности для ее применения.

Наши рассуждения опираются на статью R. Tyrrel Rockafellar, Stanislav Uryasev, "Optimization of conditional value at risk" (2000), которая в свою очередь основывается на подходе, разработанном в статье S. Uryasev, "New variable-metric algorithms for nondifferentiable optimization problems" (1991). Мы изложим технику, которую Rockafellar и Uryasev (2000) предложили для поиска оптимального портфеля, применим их метод к реальным данным и сделаем некоторые выводы. Однако мы не претендуем на авторство конкретных утверждений и результатов, а также используемых понятий из теории вероятности и оптимального управления, поэтому вся работа, сделанная лично нами, отдельно отмечена. К такой работе относятся:

- 1) Альтернативное доказательство Теоремы 2 на стр. 7.
- 2) Применение симплекс-метода к задачам P1+P2 на стр.13-14. В результате численного решения моей программой задачи линейного программирования P1+P2 были получены те же значения, что и у авторов статьи [1] в Таблицах 5 и 6.
- 3) Написание кода этой программы на языке C/C++ с использованием библиотеки glpk на стр.15-17.
- 4) Написание кода на языке Python на стр. 15 для сэмплирования нужных реализаций y , что было опущено авторами статьи [1].
- 5) Применение метода множителей Лагранжа к задаче P3 на стр.18. В результате получено такое же единственное решение, что и у авторов статьи [1] в Таблице 3.
- 6) Замечание про задачу Марковитца на максимум на стр. 20.
- 7) Написание кода на языке Python для нахождения вершин многогранника в модели Марковитца на стр. 20.
- 8) Реализация на языке C++ метода Урясьева, описанного в статье [2], и проверка совпадения теоретического и численного минимумов на стр. 23.

2. Необходимые сведения

Напомним определения таких величин, как VaR (Value at Risk) и $CVaR$ (Conditional Value at Risk), которое иначе называется хвостовым VaR . По определению, при выбранном (достаточно близком к единице) уровне доверия $\beta \in (0,1)$, $\beta\text{-}VaR$ — это такое наименьшее число α , что потери по портфелю не превысят α с вероятностью не меньше β , в то время как $\beta\text{-}CVaR$ — это условное ожидание потерь при условии превышения ими уровня α . Обычно рассматриваются три значения уровней доверия β : 0.9, 0.95 и 0.99. Из определения следует, что $\beta\text{-}VaR$ всегда не больше $\beta\text{-}CVaR$, то есть портфели с маленьким $CVaR$ имеют маленькое VaR .

Большинство подходов к оптимизации портфеля минимизируют VaR . Для этого вводятся предположения о том, что совместное распределение имеющихся на рынке активов является нормальным или логнормальным. При моделировании используются методы Монте-Карло и линейные аппроксимации портфеля.

Однако использование VaR как меры риска имеет свои недостатки, поскольку VaR не удовлетворяет условиям субаддитивности и выпуклости. Более того, Mauser и Rosen (1999), а также McKay и Keefer (1996) показали, что VaR как функция от позиций портфеля может иметь много локальных экстремумов, что является главным препятствием при попытке определения оптимальных долей каждой позиции портфеля и определения VaR каждого конкретного набора долей. Как альтернативная мера риска, $CVaR$ обладает лучшими свойствами, чем VaR . Pflug (2000) доказал, что $CVaR$ является когерентной мерой риска, обладая следующими свойствами:

- независимость от сдвига,
- однородность при умножении на положительную константу,
- выпуклость,
- монотонность.

В данной работе мы будем исследовать практическую технику, которая минимизирует $CVaR$ и вычисляет VaR одновременно. Она предоставляет удобный способ для оценки:

- линейных и нелинейных деривативов (опционы, фьючерсы),
- рыночного, кредитного и операционного рисков,
- обстоятельств в любой компании, которая подвержена финансовым рискам.

При оптимизации портфелей новый подход ведет к решению стохастической оптимизационной задачи. Для решения этой задачи существует множество численных методов. Они могут использоваться или сами по себе, или же могут быть скомбинированы с аналитическими или симуляционными методами. В случаях, когда неопределенность моделируется сценариями, и этих сценариев конечное число, задачу можно свести к одной задаче линейного программирования. Примеры применения программирования в финансах можно найти в работах Zenios (1996), а также Ziemba и Mulvey (1998).

3. Описание подхода

Пусть $f(x, y)$ — потери, связанные с вектором решений x , выбираемым из определенного подмножества $X \subseteq \mathbb{R}^n$, и случайным вектором $y \in \mathbb{R}^m$. Вектор x можно интерпретировать как портфель финансовых инструментов. Вектор y отвечает за неопределенности, например, за колебания рынка, которые могут

повлиять на потери. Конечно, потери могут быть и отрицательными, и тогда это прибыль.

Для каждого x , потери $f(x, y)$ являются случайной величиной с распределением на \mathbb{R} , индуцированным распределением y . Для удобства предполагаем, что распределение вектора $y \in \mathbb{R}^n$ имеет плотность, которую мы обозначим как $p(y)$. Однако, как будет показано в дальнейшем, для имплементации предлагаемого подхода не требуется аналитическое выражение для $p(y)$. Достаточно иметь алгоритм (программу), которая генерирует случайные величины из распределения с плотностью $p(y)$. Например, получить выборку из распределения с плотностью $p(y)$ можно с помощью метода Монте-Карло.

Вероятность того, что потери $f(x, y)$ не превысят барьер α , дается формулой:

$$\psi(x, \alpha) = \int_{y: f(x, y) \leq \alpha} p(y) dy \quad (1)$$

Как функция от α при фиксированном векторе x , функция ψ является функцией распределения потерь, ассоциированных с вектором x . Она полностью определяет поведение этой случайной величины и является фундаментальной при нахождении VaR и $CVaR$. В общем случае функция $\psi(x, \alpha)$ является неубывающей по α и непрерывной справа, но необязательно непрерывной слева из-за возможных скачков. Однако дальше мы предполагаем, что распределение вероятности такое, что скачков нет, другими словами, что функция $\psi(x, \alpha)$ везде непрерывна по α . Это предположение, как и предыдущее (про плотность вектора y), делается для упрощения, поскольку без него возникают математические сложности уже при определении $CVaR$.

Величины β - VaR и β - $CVaR$ для потерь случайной величины, ассоциированной с x и неким фиксированным уровнем доверия $\beta \in (0, 1)$, будут обозначаться как $\alpha_\beta(x)$ и $\phi_\beta(x)$. В наших обозначениях они задаются формулами:

$$\alpha_\beta(x) = \min\{\alpha \in \mathbb{R}: \psi(x, \alpha) \geq \beta\} \quad (2)$$

$$\phi_\beta(x) = \frac{1}{1 - \beta} \int_{y: f(x, y) \geq \alpha_\beta(x)} f(x, y) p(y) dy \quad (3)$$

В первой формуле $\alpha_\beta(x)$ оказывается левым концом непустого интервала, состоящего из таких значений α , что $\psi(x, \alpha) = \beta$. Это следует из того, что $\psi(x, \alpha)$ — непрерывная и неубывающая функция от α . Интервал может содержать более одной точки, если ψ имеет плоские участки.

Во второй формуле вероятность того, что $f(x, y) \geq \alpha_\beta(x)$, равна $1 - \beta$. Таким образом, $\phi_\beta(x)$ имеет смысл условного ожидания потерь, ассоциированных с вектором x , при условии, что эти потери больше или равны $\alpha_\beta(x)$.

Ключевыми в нашем подходе являются выражения для $\phi_\beta(x)$ и $\alpha_\beta(x)$ через функцию F_β , определенную на $X \times \mathbb{R}$ по следующей формуле:

$$F_\beta(x, \alpha) = \alpha + \frac{1}{1 - \beta} \int_{y \in \mathbb{R}^m} [f(x, y) - \alpha]^+ p(y) dy \quad (4)$$

Основополагающие свойства F_β отражены в Теоремах 1 и 2.

Теорема 1 (Rockafellar, Uryasev 2000)

Как функция от α , $F_\beta(x, \alpha)$ выпукла и непрерывно дифференцируема.

Более того, β -CVaR потерь, ассоциированных с произвольным $x \in X$, может быть найден из следующей формулы:

$$\phi_\beta(x) = \min_{\alpha \in \mathbb{R}} F_\beta(x, \alpha) \quad (5)$$

В этой формуле множество тех α , на которых минимум был достигнут, а именно,

$$A_\beta(x) = \operatorname{argmin}_{\alpha \in \mathbb{R}} F_\beta(x, \alpha) \quad (6)$$

является непустым отрезком (возможно, вырождающимся в одну точку), и β -VaR потерь задается формулой:

$$\alpha_\beta(x) = \text{left endpoint of } A_\beta(x). \quad (7)$$

В частности, всегда выполнено:

$$\alpha_\beta(x) \in \operatorname{argmin}_{\alpha \in \mathbb{R}} F_\beta(x, \alpha) \quad (8a)$$

$$\phi_\beta(x) = F_\beta(x, \alpha_\beta(x)) \quad (8b)$$

Отметим, что из вычислительных соображений лучше минимизировать $(1 - \beta)F_\beta(x, \alpha)$ вместо $F_\beta(x, \alpha)$. Это избавит нас от необходимости делить интеграл на маленькое число $(1 - \beta)$.

Сила формулы из Теоремы 1 очевидна, поскольку непрерывно дифференцируемые выпуклые функции очень легко численно минимизировать. Также нам открывается тот факт, что β -CVaR можно вычислить без предварительного вычисления β -VaR. Более того, β -VaR получается как побочный продукт наших вычислений, причем дополнительные усилия, которые это может повлечь за собой (вычисление интервала $A_\beta(x)$ и нахождение его левой точки, если этих точек там больше одной), могут быть опущены, если нам β -VaR не нужен.

Более того, интеграл в определении (4) для $F_\beta(x, \alpha)$ может быть приближен несколькими способами. Например, это может быть сделано путем сэмпирования вектора y в соответствии с плотностью $p(y)$. Если сэмпирование сгенерировало нам набор векторов y_1, \dots, y_q , то соответствующая аппроксимация для $F_\beta(x, \alpha)$ такова:

$$\widetilde{F}_\beta(x, \alpha) = \alpha + \frac{1}{q(1 - \beta)} \sum_{k=1}^q [f(x, y_k) - \alpha]^+ \quad (9)$$

Выражение $\widetilde{F}_\beta(x, \alpha)$ выпукло и кусочно-линейно по α . Несмотря на то, что оно не дифференцируемо по α , его легко минимизировать либо линейным поиском, либо посредством сведения к задаче линейного программирования.

Другие преимущества рассмотрения VaR и CVaR посредством формул из Теоремы 1 видны в следующей теореме.

Теорема 2 (Rockafellar, Uryasev 2000)

Минимизация β -CVaR потерь, ассоциированных с x , по множеству $x \in X$, эквивалентна минимизации $F_\beta(x, \alpha)$ по множеству $(x, \alpha) \in X \times \mathbb{R}$, в следующем смысле:

$$\min_{x \in X} \phi_\beta(x) = \min_{(x, \alpha) \in X \times \mathbb{R}} F_\beta(x, \alpha) \quad (10)$$

Здесь пара (x^*, α^*) доставляют минимум второго выражения тогда и только тогда, когда x^* доставляет минимум первого выражения, и $\alpha^* \in A_\beta(x^*)$. В частности, если $A_\beta(x^*)$ вырождается в точку, минимизация $F_\beta(x, \alpha)$ по множеству $(x, \alpha) \in X \times \mathbb{R}$ выдает пару (x^*, α^*) , не обязательно единственную, такую что x^* минимизирует β -CVaR, а α^* дает соответствующее β -VaR.

Более того, когда $f(x, y)$ выпукла по x , то $F_\beta(x, \alpha)$ выпукла по (x, α) и $\phi_\beta(x)$ выпукла по x , а если еще X является выпуклым множеством, то задача минимизации является задачей выпуклого программирования.

Отметим, что лично нами было придумано альтернативное доказательство Теоремы 2, которое нам кажется очень понятным и логичным. Для большей наглядности мы будем проводить доказательство для одномерного y .

3.1. Альтернативное доказательство Теоремы 2

Итак, мы хотим обосновать, почему β -CVaR (то есть $\phi_\beta(x)$) можно находить не по определению (см. стр 5), а как минимальное значение следующего выражения, причем минимум достигается при $\alpha = \alpha_\beta(x)$ (то есть $\alpha = \beta$ -VaR):

$$F_\beta(x, \alpha) = \alpha + \frac{1}{1-\beta} \int_{-\infty}^{+\infty} [f(x, y) - \alpha]^+ p(y) dy = \alpha + \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha}^{+\infty} [f(x, y) - \alpha] p(y) dy$$

Для этого преобразуем определение $\phi_\beta(x)$ к следующему виду:

$$\begin{aligned} \phi_\beta(x) &= \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} f(x, y) p(y) dy = \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} [f(x, y) \pm \alpha] p(y) dy = \\ &= \frac{1}{1-\beta} \cdot \alpha \cdot \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} p(y) dy + \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} [f(x, y) - \alpha] p(y) dy = \\ &= \frac{1}{1-\beta} \cdot \alpha \cdot P(f(x, y) \geq \alpha_\beta(x)) + \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} [f(x, y) - \alpha] p(y) dy = \\ &= \frac{1}{1-\beta} \cdot \alpha \cdot (1-\beta) + \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} [f(x, y) - \alpha] p(y) dy = \\ &= \alpha + \frac{1}{1-\beta} \int_{y: f(x, y) \geq \alpha_\beta(x)}^{+\infty} [f(x, y) - \alpha] p(y) dy \end{aligned}$$

Теперь сравним преобразованное определение $\phi_\beta(x)$ и функцию $F_\beta(x, \alpha)$.

- При $\alpha = \alpha_\beta(x)$: эти выражения в точности равны.
- При $\alpha > \alpha_\beta(x)$: у преобразованного определения $\phi_\beta(x)$ есть часть при $y \in [\alpha_\beta(x), \alpha]$, когда подынтегральное выражение отрицательно, а в формуле для $F_\beta(x, \alpha)$ это слагаемое выброшено, значит, $\phi_\beta(x) < F_\beta(x, \alpha)$.
- При $\alpha < \alpha_\beta(x)$: и у преобразованного определения $\phi_\beta(x)$, и у $F_\beta(x, \alpha)$ подынтегральное выражение всегда положительно, но в формуле для $F_\beta(x, \alpha)$ область интегрирования больше, значит, $\phi_\beta(x) < F_\beta(x, \alpha)$.

Получаем, что минимальное значение $F_\beta(x, \alpha)$ в точности равно $\phi_\beta(x)$, причем минимум достигается при $\alpha = \alpha_\beta(x)$, что и утверждает теорема.

То есть мы доказали, что при фиксированном x : $\phi_\beta(x) = \min_{\alpha} F_\beta(x, \alpha)$.

Значит, если разрешено варьировать x , то $\min_x \phi_\beta(x) = \min_{x, \alpha} F_\beta(x, \alpha)$.

Другими словами, если мы хотим найти минимум $\phi_\beta(x)$ по портфелям x , то действительно нужно найти минимум функции $F_\beta(x, \alpha)$ по переменным (x, α) . Теорема доказана!

Итак, согласно Теореме 2, для задачи нахождения минимизирующего β -CVaR вектора x необязательно работать непосредственно с функцией $\phi_\beta(x)$, которую сложно минимизировать в силу ее определения как β -VaR-значения $\alpha_\beta(x)$ и часто неудобных математических свойств $\alpha_\beta(x)$. Вместо этого, мы можем оперировать с намного более простым выражением $F_\beta(x, \alpha)$, которое выпукло по α и очень часто выпукло по (x, α) .

Метод оптимизации, основанный на Теореме 2, может быть скомбинирован с разными идеями аппроксимации интеграла в определении (4) для $F_\beta(x, \alpha)$. Это предоставляет широкие возможности. Например, выпуклость $f(x, y)$ по x влечет выпуклость аппроксимирующего выражения $\widetilde{F}_\beta(x, \alpha)$ в (9).

Задача минимизации $F_\beta(x, \alpha)$ по множеству $X \times \mathbb{R}$ является задачей стохастической оптимизации, или, более строго, задачей стохастического программирования, из-за наличия математического ожидания в определении $F_\beta(x, \alpha)$. Как минимум в случаях выпуклости, существует огромное количество способов решения задачи. Теорема 2 предоставляет возможность для применения подходов стохастического программирования к задаче минимизации β -CVaR.

4. Применение к оптимизации портфеля

Для иллюстрации предлагаемого нами подхода, рассмотрим случай, когда вектор решений x соответствует финансовым инструментам в том смысле, что $x = (x_1, \dots, x_n)$, где x_j — это позиция по инструменту j , и

$$\begin{cases} x_j \geq 0, j = 1, \dots, n \\ \sum_{j=1}^n x_j = 1 \end{cases} \quad (11)$$

Обозначая за y_j прибыль по инструменту j , мы получаем случайный вектор $y = (y_1, \dots, y_n)$.

Такое описание y составляет совместное распределение различных прибылей и не зависит от x . Совместное распределение имеет плотность $p(y)$.

Прибыль по портфелю x — это сумма прибылей по каждому инструменту портфеля, взятая с коэффициентом x_j . Тогда потери — это такое же выражение, но с обратным знаком:

$$f(x, y) = -[x_1 y_1 + \dots + x_n y_n] = -x^T y \quad (12)$$

Поскольку $p(y)$ непрерывна по y , то функция распределения потерь, соответствующая вектору x , сама тоже будет непрерывна (см. Кап и Kibzun (1996), Uruasev (1995)).

Несмотря на то, что VaR и CVaR обычно имеют размерность денег, у нас они имеют размерность процентного дохода. Мы рассматриваем случай, когда задано взаимно-однозначное соответствие между процентным доходом и численным значением в деньгах. В этом разделе мы сравним методологию минимизации CVaR с методологией минимизации дисперсии, поэтому мы и рассматриваем потери в терминах процентов.

Мы хотим минимизировать функцию

$$F_\beta(x, \alpha) = \alpha + \frac{1}{1-\beta} \int_{y \in \mathbb{R}^n} [-x^T y - \alpha]^+ p(y) dy \quad (13)$$

Важно отметить, что в такой постановке функция $F_\beta(x, \alpha)$ выпукла по паре аргументов (x, α) , а не только по α . Часто она еще и дифференцируема по этим аргументам (см. Кан и Kibzun (1996), Uruasev (1995)). Это свойство делает привлекательным применение вышеназванных численных методов оптимизации.

Для портфеля x обозначим $\mu(x)$ и $\sigma(x)$ математическое ожидание и дисперсию потерь, соответственно. В терминах математического ожидания m и дисперсии V вектора y , мы имеем:

$$\mu(x) = -x^T m \quad (14a)$$

$$\sigma(x) = x^T V x \quad (14b)$$

Формулы имеют место, поскольку $\mu(x)$ — линейная функция от x , а $\sigma(x)$ — квадратичная функция от x . Мы налагаем требование, что допускаются только портфели, от которых можно ожидать возврата хотя бы данного значения R . Другими словами, мы налагаем линейное ограничение:

$$\mu(x) \leq -R \quad (15)$$

и допустимыми считаем только такие портфели:

$$X = \{x: x \text{ удовлетворяет (11) и (15)}\} \quad (16)$$

Множество X выпукло (потому что это многогранник, в силу линейности ограничений). Поэтому задача минимизации $F_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$ является задачей выпуклого программирования.

Теперь рассмотрим вид (9) аппроксимации функции $F_\beta(x, \alpha)$, получаемый из сэмплирования векторов из распределения y :

$$\widetilde{F}_\beta(x, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^q [f(x, y_k) - \alpha]^+ \quad (17)$$

Задача минимизация $\widetilde{F}_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$ для нахождения приближенного решения задачи минимизации $F_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$ может быть сведена к задаче выпуклого программирования. В терминах вспомогательных действительных величин $u_k, k = 1, \dots, q$, эта задача эквивалентна минимизации линейного выражения:

$$\alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^q u_k$$

при условии выполнении линейных ограничений:

$$\left\{ \begin{array}{l} u_k \geq 0, k = 1, \dots, q \\ x_i \geq 0, i = 1, 2, 3 \\ \alpha \geq 0 \\ (x^T y)_k + \alpha + u_k \geq 0, k = 1, \dots, q \\ x_1 + x_2 + x_3 = 1 \\ x_1 m_1 + x_2 m_2 + x_3 m_2 \geq R \end{array} \right.$$

Отметим, что возможность такого сведения в задаче линейного программирования не зависит от распределения вектора y , то есть метод будет

работать даже в случае, если вектор y имеет распределение, отличное от нормального.

Наш анализ до сих пор был направлен на минимизацию β -CVaR, иными словами, на задачу (P1):

$$(P1) \quad \text{minimize } \phi_{\beta}(x) \text{ на множестве } x \in X,$$

поскольку это и происходит (согласно Теореме 2) при минимизации $F_{\beta}(x, \alpha)$ на множестве $X \times \mathbb{R}$. Схожая проблема поиска портфеля, минимизирующего β -VaR, есть решение задачи

$$(P2) \quad \text{minimize } \alpha_{\beta}(x) \text{ на множестве } x \in X,$$

полностью не покрывается нашей Теоремой 2. Но поскольку $\phi_{\beta}(x) \geq \alpha_{\beta}(x)$, то решения задачи (P1) хороши с точки зрения задачи (P2). В силу Теоремы 2, техника минимизации $F_{\beta}(x, \alpha)$ на множестве $X \times \mathbb{R}$ для решения задачи (P1) также выдает β -VaR портфеля x^* , минимизирующего β -CVaR. Это не то же самое, что решение задачи (P2), но отсюда видно, что менеджменту выгоднее решать задачу (P1), чем задачу (P2).

Также полезно сравнить задачи (P1) и (P2) с очень известной задачей минимизации дисперсии:

$$(P3) \quad \text{minimize } \sigma^2(x) \text{ на множестве } x \in X.$$

Хорошим математическим свойством задачи (P3) является то, что она сводится к задаче квадратичного программирования, но, как и в случае с задачей (P2), применимость решения этой задачи под вопросом. Также можно отметить несколько других подходов: подход минимизации абсолютного отклонения и минимаксный подход достойны внимания в связи с аппроксимационной схемой (17) для минимизации CVaR, поскольку эти подходы тоже являются подходами линейного программирования.

Эти задачи могут выдать, по крайней мере в одном важном случае, один и тот же оптимальный портфель x^* . Далее мы сформулируем этот факт и проверим его на практике.

Предложение 1

Предположим, что потери, соответствующие портфелю при каждом векторе решений x , имеют нормальное распределение (это имеет место, если вектор y имеет нормальное распределение). Если $\beta \geq 0.5$ и на решениях любых двух задач (P1), (P2), (P3) в ограничении (15) достигается равенство, то решения этих двух задач совпадают. То есть общий портфель x^* является оптимальным по обоим критериям.

Доказательство:

В предположении нормальности распределения и при $\beta \geq 0.5$, выразим β -VaR и β -CVaR через математическое ожидание и дисперсию:

$$\alpha_{\beta}(x) = \mu(x) + c_1(\beta)\sigma(x), \text{ где } c_1(\beta) = \sqrt{2}erf^{-1}(2\beta - 1) \quad (18)$$

$$\phi_{\beta}(x) = \mu(x) + c_2(\beta)\sigma(x), \text{ где } c_2(\beta) = \left\{ \sqrt{2\pi} \exp[erf^{-1}(2\beta - 1)]^2 (\beta - 1) \right\}^{-1} \quad (19)$$

Здесь erf^{-1} обозначает функцию, обратную к

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

Когда в ограничении (15) на решении достигается равенство, то множество X может быть заменено при минимизации на меньшее множество X' , получаемое заменой неравенства $\mu(x) \leq -R$ на равенство $\mu(x) = -R$. Но для X' мы имеем:

$$\alpha_\beta(x) = -R + c_1(\beta)\sigma(x),$$

$$\phi_\beta(x) = -R + c_2(\beta)\sigma(x),$$

Причем коэффициенты $c_1(\beta)$ и $c_2(\beta)$ положительны. Поэтому очевидно, что минимизация этих выражений по множеству X' эквивалентна минимизации $\sigma^2(x)$ на множестве $x \in X'$.

Таким образом, если в ограничении (15) достигается равенство на решении двух задач, то любой портфель x^* , минимизирующий $\sigma(x)$ на множестве $x \in X'$, является оптимальным для этих двух задач.

Предложение 1 предлагает возможность использования решений задачи (P3), полученных квадратичным программированием, в качестве ориентира при тестировании метода оптимизации β -CVaR путем сэмплирования аппроксимаций (17) и сведения их к задаче линейного программирования. Мы осуществляем это на примере, когда нужно составить портфель из S&P500, долгосрочных правительственных облигаций США и мелких ценных бумаг, а доходы по этим инструментам имеют совместное нормальное распределение.

Среднее значение m месячных выплат и ковариационная матрица V для этого примера даны в Таблицах 1 и 2, соответственно. Мы взяли $R = 0.011$ в качестве ограничения (15).

| | |
|-----------|-----------|
| S&P | 0.0101110 |
| Gov. bond | 0.0043532 |
| Small cap | 0.0137058 |

Таблица 1. Средние доходности инструментов портфеля.

| | S&P | Gov. bond | Small cap |
|-----------|------------|------------|------------|
| S&P | 0.00324625 | 0.00022983 | 0.00420395 |
| Gov. bond | 0.00022983 | 0.00049937 | 0.00019247 |
| Small cap | 0.00420395 | 0.00019247 | 0.00764097 |

Таблица 2. Матрица ковариаций инструментов портфеля.

Рассмотрим сначала, какие результаты были получены в работе [1], а потом сравним их с тем, что получилось у нас.

5. Результаты работы [1] для рассматриваемой задачи оптимизации портфеля

Сначала авторы статьи решили задачу квадратичного программирования (P3) для представленных выше данных. В результате получился портфель x^* , показанный в Таблице 3, который является единственным оптимальным портфелем в смысле минимальности дисперсии по Марковитцу. Соответствующая дисперсия получилась равной $\sigma^2(x^*) = 0.00378529$, а среднее $\mu(x^*) = -0.011$. Таким образом, в ограничении (15) для задачи (P3) достиглось равенство.

| S&P | Gov. bond | Small cap |
|----------|-----------|-----------|
| 0.452013 | 0.115573 | 0.432414 |

Таблица 3. Оптимальный состав портфеля в задаче минимизации VaR .

Далее, для значений $\beta = 0.99, 0.95, 0.9$, были вычислены β - VaR и β - $CVaR$ портфеля x^* по формулам (18) и (19), полученные результаты занесены в таблицу 4.

| | $\beta = 0.90$ | $\beta = 0.95$ | $\beta = 0.99$ |
|------|----------------|----------------|----------------|
| VaR | 0.067847 | 0.090200 | 0.132128 |
| CVaR | 0.096975 | 0.115908 | 0.152977 |

Таблица 4. VaR и $CVaR$ для оптимального портфеля в задаче минимизации VaR .

Имея эти значения для последующего сравнения, далее авторы применили свой подход для решения задачи (P1) минимизации $F_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$, основанный на Теореме 2. Для аппроксимации интеграла в выражении (13) для $F_\beta(x, \alpha)$, они (не указывая способа сэмпирования) насэмплировали реализации вектора y из многомерного нормального распределения $\mathcal{N}(m, V) \in \mathbb{R}^3$. Эти реализации вектора y предоставили нам аппроксимацию $\widetilde{F}_\beta(x, \alpha)$ из (17). Задача минимизации $\widetilde{F}_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$ в каждом случае была сведена к задаче линейного программирования способом, описанным после (17). Эти вычисления выдали оценку x^* оптимального портфеля в задаче (P1), вместе с соответствующими оценками α^* для β - VaR и $F_\beta(x, \alpha^*)$ для β - $CVaR$.

Вычисления при решении задачи линейного программирования проводились с помощью метода CPLEX на машине Пентиум 2. При генерации случайных величин использовались два вида “случайных” чисел: обычный метода Монте-Карло и метод с использованием Соболевских псевдослучайных чисел. Результаты для первого типа псевдослучайных чисел показаны в Таблице 5, а для второго типа — в Таблице 6.

| β | Sample size | S&P | Gov. bond | Small cap | VaR | VaR diff. (%) | CVaR | CVaR diff. (%) | Iterations | Time (min) |
|---------|-------------|---------|-----------|-----------|---------|---------------|---------|----------------|------------|------------|
| 0.9 | 1000 | 0.35250 | 0.15382 | 0.49368 | 0.06795 | 0.154 | 0.09962 | 2.73 | 1157 | 0.0 |
| 0.9 | 3000 | 0.55726 | 0.07512 | 0.36762 | 0.06537 | 3.645 | 0.09511 | -1.92 | 636 | 0.0 |
| 0.9 | 5000 | 0.42914 | 0.12436 | 0.44649 | 0.06662 | 1.809 | 0.09824 | 1.30 | 860 | 0.1 |
| 0.9 | 10000 | 0.48215 | 0.10399 | 0.41386 | 0.06622 | 2.398 | 0.09503 | -2.00 | 2290 | 0.3 |
| 0.9 | 20000 | 0.45951 | 0.11269 | 0.42780 | 0.06629 | -2.299 | 0.09602 | -0.98 | 8704 | 1.5 |
| 0.95 | 1000 | 0.53717 | 0.08284 | 0.37999 | 0.09224 | 2.259 | 0.11516 | -0.64 | 156 | 0.0 |
| 0.95 | 3000 | 0.54875 | 0.07839 | 0.37286 | 0.09428 | 4.524 | 0.11888 | 2.56 | 652 | 0.0 |
| 0.95 | 5000 | 0.57986 | 0.06643 | 0.35371 | 0.09175 | 1.715 | 0.11659 | 0.59 | 388 | 0.1 |
| 0.95 | 10000 | 0.47102 | 0.10827 | 0.42072 | 0.08927 | -1.03 | 0.11467 | -1.00 | 1451 | 0.2 |
| 0.95 | 20000 | 0.49038 | 0.10082 | 0.40879 | 0.09136 | 1.284 | 0.11719 | 1.11 | 2643 | 0.7 |
| 0.99 | 1000 | 0.41844 | 0.12848 | 0.45308 | 0.13454 | 1.829 | 0.14513 | -5.12 | 340 | 0.0 |
| 0.99 | 3000 | 0.61960 | 0.05116 | 0.32924 | 0.12791 | -3.187 | 0.14855 | -2.89 | 1058 | 0.0 |
| 0.99 | 5000 | 0.63926 | 0.04360 | 0.31714 | 0.13176 | -0.278 | 0.15122 | -1.14 | 909 | 0.1 |
| 0.99 | 10000 | 0.45203 | 0.11556 | 0.43240 | 0.12881 | -2.51 | 0.14791 | -3.31 | 680 | 0.1 |
| 0.99 | 20000 | 0.45766 | 0.11340 | 0.42894 | 0.13153 | -0.451 | 0.15334 | 0.24 | 3083 | 0.9 |

Таблица 5. Состав оптимального портфеля, его VaR и $CVaR$ в задаче минимизации $CVaR$ с использованием псевдослучайных чисел.

| β | Sample size | S&P | Gov. bond | Small cap | VaR | VaR diff. (%) | CVaR | CVaR diff. (%) | Iterations | Time (min) |
|---------|-------------|---------|-----------|-----------|---------|---------------|---------|----------------|------------|------------|
| 0.9 | 1000 | 0.43709 | 0.12131 | 0.44160 | 0.06914 | 1.90 | 0.09531 | -1.71 | 429 | 0.0 |
| 0.9 | 3000 | 0.45425 | 0.11471 | 0.43104 | 0.06762 | -0.34 | 0.09658 | -0.41 | 523 | 0.0 |
| 0.9 | 5000 | 0.44698 | 0.11751 | 0.43551 | 0.06784 | -0.02 | 0.09664 | -0.35 | 837 | 0.1 |
| 0.9 | 10000 | 0.45461 | 0.11457 | 0.43081 | 0.06806 | 0.32 | 0.09695 | -0.02 | 1900 | 0.3 |
| 0.9 | 20000 | 0.46076 | 0.11221 | 0.42703 | 0.06790 | 0.08 | 0.09692 | -0.06 | 4818 | 0.6 |
| 0.95 | 1000 | 0.43881 | 0.12065 | 0.44054 | 0.09001 | -0.21 | 0.11249 | -2.95 | 978 | 0.0 |
| 0.95 | 3000 | 0.43881 | 0.12065 | 0.44054 | 0.09001 | -0.21 | 0.11511 | -0.69 | 407 | 0.0 |
| 0.95 | 5000 | 0.46084 | 0.11218 | 0.42698 | 0.09036 | 0.18 | 0.11516 | -0.64 | 570 | 0.1 |
| 0.95 | 10000 | 0.45723 | 0.11357 | 0.42920 | 0.09016 | -0.05 | 0.11577 | -0.12 | 1345 | 0.2 |
| 0.95 | 20000 | 0.45489 | 0.11447 | 0.43064 | 0.09023 | 0.03 | 0.11577 | -0.12 | 1851 | 0.7 |
| 0.99 | 1000 | 0.52255 | 0.08846 | 0.38899 | 0.12490 | -5.47 | 0.14048 | -8.17 | 998 | 0.0 |
| 0.99 | 3000 | 0.43030 | 0.12392 | 0.44578 | 0.12801 | -3.12 | 0.15085 | -1.39 | 419 | 0.0 |
| 0.99 | 5000 | 0.45462 | 0.11457 | 0.43081 | 0.13073 | -1.06 | 0.14999 | -1.95 | 676 | 0.1 |
| 0.99 | 10000 | 0.39156 | 0.13881 | 0.46963 | 0.13288 | 0.57 | 0.15208 | -0.59 | 1065 | 0.2 |
| 0.99 | 20000 | 0.46065 | 0.11225 | 0.42710 | 0.13198 | -0.11 | 0.15211 | -0.57 | 1317 | 0.5 |

Таблица 6. Состав оптимального портфеля, его VaR и $CVaR$ в задаче минимизации $CVaR$ с использованием Соболевских квазислучайных чисел.

Сравнивая результаты из Таблиц 5 и 6 для подхода $CVaR$ с результатами подхода минимальной дисперсии из Таблиц 3 и 4, мы видим, что значения $CVaR$ отличаются всего на несколько процентов, в зависимости от размера выборки, и аналогичное верно для VaR . Однако сходимость значений $CVaR$ из Таблиц 5 и 6 к значениям Таблицы 4 (что должно иметь место согласно Предложению 1) довольно медленная. Это может быть связано с погрешностями сэмпирования при симуляции Монте-Карло. Кроме того, оказалось, что в случае оптимальности портфеля, его дисперсия, VaR и $CVaR$ слабо зависят от изменений позиций портфеля.

6. Мои результаты в задаче оптимизации портфеля

В главе 3 поставлены три задачи $P1+P2$ и $P3$, причем $P1+P2$ является линейной, а $P3$ — выпуклой, и утверждается, что при условии гауссовости распределения y , решения задач $P1+P2$ и $P3$ будут совпадать. Напишем для задач $(P1+P2)$ программу на языке C, которая будет находить решение задачи линейного программирования $(P1+P2)$ симплекс-методом с помощью библиотеки `glpk-5.0`, и сравним полученные нами результаты с результатами в Таблицах 5 и 6. Кроме того, применим теорему Каруша-Куна-Таккера к выпуклой задаче $P3$, методом множителей Лагранжа найдем оптимальное значение вектора долей соответствующих инструментов $x = (x_1, x_2, x_3)$, убедимся, что оптимальное значение действительно единственно и сравним полученные x , VaR , $CVaR$ с результатами из Таблицы 3.

6.1. Реализация симплекс-метода `glpk` для задачи $P1+P2$

Мы предполагаем, что вектор $y = (y_1, y_2, y_3)$, соответствующий ценам трех исследуемых финансовых инструментов, имеет многомерное нормальное распределение с вектором математических ожиданий m и матрицей ковариаций cov , которые заданы в Таблицах 1 и 2 соответственно.

| | |
|-----------|-----------|
| S&P | 0.0101110 |
| Gov. bond | 0.0043532 |
| Small cap | 0.0137058 |

Таблица 6. Средние доходности инструментов портфеля.

| | S&P | Gov. bond | Small cap |
|-----------|------------|------------|------------|
| S&P | 0.00324625 | 0.00022983 | 0.00420395 |
| Gov. bond | 0.00022983 | 0.00049937 | 0.00019247 |
| Small cap | 0.00420395 | 0.00019247 | 0.00764097 |

Таблица 7. Матрица ковариаций инструментов портфеля.

Наша цель — минимизировать VaR и CVaR портфеля, что, согласно Теореме 1, можно сделать путем минимизации функции:

$$F_{\beta}(x, \alpha) = \alpha + \frac{1}{1-\beta} \int_{y \in \mathbb{R}^m} [f(x, y) - \alpha]^+ p(y) dy \quad (4)$$

Как уже говорилось выше, интеграл в определении (4) для $F_{\beta}(x, \alpha)$ может быть приближен несколькими способами. Например, это может быть сделано путем сэмпирования вектора y в соответствии с плотностью $p(y)$. Если сэмпирование сгенерировало нам набор векторов y_1, \dots, y_q , то соответствующая аппроксимация для $F_{\beta}(x, \alpha)$ такова:

$$\widetilde{F}_{\beta}(x, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^q [f(x, y_k) - \alpha]^+ \quad (9)$$

В терминах вспомогательных действительных величин $u_k, k = 1, \dots, q$, эта задача эквивалентна минимизации линейного выражения:

$$\alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^q u_k$$

при условии выполнении линейных ограничений:

$$\begin{cases} u_k \geq 0, k = 1, \dots, q \\ x_i \geq 0, i = 1, 2, 3 \\ \alpha \geq 0 \\ (x^T y)_k + \alpha + u_k \geq 0, k = 1, \dots, q \\ x_1 + x_2 + x_3 = 1 \\ x_1 m_1 + x_2 m_2 + x_3 m_2 \geq R \end{cases}$$

Сэмпирование q штук реализаций вектора $y = (y_1, y_2, y_3)$ из многомерного нормального распределения с вектором математических ожиданий m и матрицей ковариаций cov реализовано на языке Python3 в файле `generatey1y2y3.py` (число q передается при запуске этой программы в качестве аргумента командной строки), в результате чего полученная выборка записывается в файл `Generatedy1y2y3.txt`.

```

1 import numpy as np
2 import sys
3
4 q=int(sys.argv[1])
5 mean=np.array([0.010111,0.0043552,0.0137058])
6 cov=np.array([[0.00324625,0.00022983,0.00420395],
7               [0.00022983,0.00049937,0.00019247],
8               [0.00420395,0.00019247,0.00764897]])
9
10 otv=np.random.multivariate_normal(mean,cov,q);
11 np.savetxt('Generatedy1y2y3.txt',otv)
12

```

Рис. 1 Файл generatey1y2y3.py

Симплекс-метод нахождения оптимального портфеля $x = (x_1, x_2, x_3)$ находится в файле ProblemsP1P2Glpk.cpp, но для того, чтобы этот файл запустился, нужно скачать библиотеку glpk-5.0. Для этого нужно зайти на сайт <https://www.gnu.org/software/glpk/> и нажать в верхнем горизонтальном меню Downloading, потом на открывшейся странице под заголовком Downloading GLPK найти ссылку <http://ftp.gnu.org/gnu/glpk/> и по ней скачать архив [glpk-5.0.tar.gz](http://ftp.gnu.org/gnu/glpk/glpk-5.0.tar.gz). Потом его надо разархивировать в какую-нибудь папку, в этой папке зайти в подпапку glpk-5.0, там будет файл INSTALL, и там написано, что сначала надо в командной строки из этой подпапки glpk-5.0 запустить команду ./configure, потом make, потом make install. После этого библиотека glpk скачана, и ее можно включать в любой c++-файл командой #include<glpk.h>. Посмотрим на программу ProblemsP1P2Glpk.cpp:

```

1 #include<iostream>
2 #include<fstream>
3 #include<glpk.h>
4
5 void push(int* ia, int* ja, double* arr, int i, int j, double value, int& next){
6     ia[next]=i;
7     ja[next]=j;
8     arr[next]=value;
9     ++next;
10 }
11
12 int main(int argc, char** argv){
13     if (argc!=3){std::cout<<"Usage: ./a.out q=10000 b=0.9"<<std::endl; return 0;}
14     glp_prob* lp_solver;
15     glp_smcp solver_params;
16
17     int q=atoi(argv[1]); // 10000 num of generated (y1,y2,y3)
18     double beta=atof(argv[2]);
19     double R=0.011;
20     int* ia = (int*)malloc(((q+4)*(q+4)+1)*sizeof(int)); //masssiv for i-index of nonzero elements in matrix a
21     int* ja = (int*)malloc(((q+4)*(q+4)+1)*sizeof(int)); //masssiv for j-index of nonzero elements in matrix a
22     double* arr=(double*)malloc(((q+4)*(q+4)+1)*sizeof(double)); //massiv for values of nonzero elements in matrix a
23
24     lp_solver=glp_create_prob();
25     glp_set_prob_name(lp_solver, "Problem P1P2 linear solver");
26     glp_set_obj_dir(lp_solver, GLP_MIN); //we want to minimize the functional
27
28     //add rows means add linear constraints
29     glp_add_rows(lp_solver, q+2); //we have q+2 constraints; (u,x,alpha)
30     for (int i=1; i<=q; i=i+1){
31         glp_set_row_bnds(lp_solver, i, GLP_LO, 0.0, 0.0); //xTy_k+alpha+uk >=0 lhs<=...<infinity
32     }
33     glp_set_row_bnds(lp_solver, q+1, GLP_FX, 1.0, 1.0); //x1+x2+x3=1
34     glp_set_row_bnds(lp_solver, q+2, GLP_LO, R, R); //xTm>=R
35
36
37
38     //add cols means add variables u, x, alpha

```



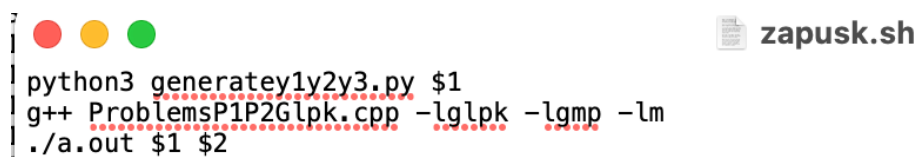
```

37
38 //add cols means add variables u,x,alpha
39 glp_add_cols(lp_solver, q+3+1); //u,x,alpha
40 for (int i=1; i<=q; i=i+1){
41     glp_set_col_bnds(lp_solver, i, GLP_LO, 0.0, 0.0); //u_k>=0
42 }
43 for (int i=1; i<=3; i=i+1){
44     glp_set_col_bnds(lp_solver, q+i, GLP_LO, 0.0, 0.0); //x_i>=0
45 }
46 glp_set_col_bnds(lp_solver, q+3+1, GLP_FR, 0.0, 0.0); //x_i>=0
47
48
49
50 //fill coef of target functional
51 for (int i=1; i<=q; i=i+1){
52     glp_set_obj_coef(lp_solver, i, 1/((double)(q)*(1-beta))); //coef of u_k in functional are 1/(q(1-beta))
53 }
54 for (int i=1; i<=3; i=i+1){
55     glp_set_obj_coef(lp_solver, q+i, 0.0); //coef of x_i in functional are 0
56 }
57 glp_set_obj_coef(lp_solver, q+3+1, 1.0); //coef of alpha in functional
58
59 int next=1; //numeration of nonzero elements is from 1 (not from 0)
60 std::ifstream in("Generatedy1y2y3.txt");
61 for (int i=1; i<=q; i=i+1){
62     double y1,y2,y3;
63     in>>y1>>y2>>y3;
64     push(ia,ja,arr,i,q+1,y1,next); //coef y1 for x1 in xTy_k+alpha+uk >=0
65     push(ia,ja,arr,i,q+2,y2,next); //coef y2 for x2 in xTy_k+alpha+uk >=0
66     push(ia,ja,arr,i,q+3,y3,next); //coef y3 for x3 in xTy_k+alpha+uk >=0
67     push(ia,ja,arr,i,q+4,1.0,next); //coef 1 for alpha in xTy_k+alpha+uk >=0
68     push(ia,ja,arr,i,i,1.0,next); //coef 1 for uk in xTy_k+alpha+uk >=0
69 }
70 push(ia,ja,arr,q+1,q+1,1,next);
71 push(ia,ja,arr,q+1,q+2,1,next);
72 push(ia,ja,arr,q+1,q+3,1,next);
73
74 double m1=0.01000111;
75 double m2=0.0043532;
76 double m3=0.0137058;
77
78 push(ia,ja,arr,q+2,q+1,m1,next);
79 push(ia,ja,arr,q+2,q+2,m2,next);
80 push(ia,ja,arr,q+2,q+3,m3,next);
81
82 glp_load_matrix(lp_solver, next-1, ia, ja, arr);
83
84 solver_params.msg_lev=GLP_MSG_ALL;
85 solver_params.meth=GLP_PRIMAL; //we solve normal, not inverse problem
86 glp_init_smcp(&solver_params);
87 glp_simplex(lp_solver, &solver_params);
88
89 double cvar=glp_get_obj_val(lp_solver);
90 double var=glp_get_col_prim(lp_solver, q+3+1);
91
92 double x1=glp_get_col_prim(lp_solver, q+1);
93 double x2=glp_get_col_prim(lp_solver, q+2);
94 double x3=glp_get_col_prim(lp_solver, q+3);
95
96 std::cout<<beta<<"-beta-CVAR="<<cvar<<std::endl;
97 std::cout<<beta<<"-beta-VAR="<<var<<std::endl;
98 std::cout<<"x1="<<x1<<std::endl;
99 std::cout<<"x2="<<x2<<std::endl;
100 std::cout<<"x3="<<x3<<std::endl;
101 std::cout<<beta<<" '<<q<<' '<<x1<<' '<<x2<<' '<<x3<<' '<<var<<' '<<cvar<<std::endl;
102
103 glp_delete_prob(lp_solver);
104
105 free(ia);
106 free(ja);
107 free(arr);
108
109 return 0;
110 }

```

Рис. 2 Программа-реализация симплекс-метода в файле ProblemsP1P2Glpk.cpp

Также написан скрипт `zapusk.sh`, который принимает на вход q и β в качестве аргумента командной строки, сам запускает `python3`-файл для генерации выборки векторов $y = (y_1, y_2, y_3)$ размера q , с этой выборкой запускает программу `ProblemsP1P2Glpk.cpp` и получает ответ: значения $\beta, q, x_1, x_2, x_3, VaR, CVaR$.



```
python3 generatey1y2y3.py 10000
g++ ProblemsP1P2Glpk.cpp -lglpk -lgmp -lm
./a.out 10000 0.9
```

Рис. 3 Скрипт `zapusk.sh` для запуска `generatey1y2y3.py` и `ProblemsP1P2Glpk.cpp`.

Например, запустим программу с параметрами $\beta = 0.9, q = 10000$:

```
((base) MacBook-Pro-Aleksandra:glpk aleksandra$ sh zapusk.sh 10000 0.9
GLPK Simplex Optimizer 5.0
10002 rows, 10004 columns, 50006 non-zeros
  0: obj = 0.000000000e+00 inf = 1.011e+00 (2)
 121: obj = 1.353120338e-01 inf = 0.000e+00 (0) 1
* 1016: obj = 9.725330346e-02 inf = 0.000e+00 (0) 9
OPTIMAL LP SOLUTION FOUND
0.9-beta-CVAR=0.0972533
0.9-beta-VAR=0.0680237
x1=0.42077
x2=0.122637
x3=0.456592
0.9 10000 0.42077 0.122637 0.456592 0.0680237 0.0972533
```

Рис. 4 Пример запуска программы с параметрами $\beta = 0.9, q = 10000$.

Получили значения, близкие к значениям Таблиц 5 и 6 для соответствующих β, q . Выполним запуск для $\beta = 0.9, 0.95, 0.99$ и $q = 1000, 3000, 5000, 10000, 20000$, занесем результаты в таблицу, представленную на Рис. 5:

| | beta | q | x1 | x2 | x3 | Var | Cvar |
|----|------|-------|----------|----------|----------|----------|----------|
| 0 | 0.90 | 1000 | 0.313049 | 0.165307 | 0.521644 | 0.063062 | 0.094807 |
| 1 | 0.90 | 3000 | 0.419921 | 0.122974 | 0.457106 | 0.064191 | 0.093045 |
| 2 | 0.90 | 5000 | 0.468433 | 0.103758 | 0.427810 | 0.070526 | 0.099258 |
| 3 | 0.90 | 10000 | 0.420770 | 0.122637 | 0.456592 | 0.068024 | 0.097253 |
| 4 | 0.90 | 20000 | 0.405869 | 0.128540 | 0.465591 | 0.068102 | 0.096490 |
| 5 | 0.95 | 1000 | 0.445560 | 0.112818 | 0.441622 | 0.090979 | 0.114206 |
| 6 | 0.95 | 2000 | 0.361934 | 0.145943 | 0.492123 | 0.087965 | 0.113116 |
| 7 | 0.95 | 3000 | 0.356371 | 0.148146 | 0.495482 | 0.093990 | 0.119796 |
| 8 | 0.95 | 5000 | 0.384653 | 0.136944 | 0.478403 | 0.089157 | 0.114562 |
| 9 | 0.95 | 10000 | 0.308896 | 0.166952 | 0.524152 | 0.091168 | 0.118805 |
| 10 | 0.95 | 20000 | 0.355270 | 0.148583 | 0.496147 | 0.089720 | 0.114607 |
| 11 | 0.99 | 1000 | 0.393467 | 0.133453 | 0.473081 | 0.119341 | 0.131075 |
| 12 | 0.99 | 3000 | 0.542996 | 0.074222 | 0.382782 | 0.139283 | 0.156592 |
| 13 | 0.99 | 5000 | 0.394506 | 0.133041 | 0.472453 | 0.141016 | 0.163303 |
| 14 | 0.99 | 10000 | 0.309204 | 0.166830 | 0.523966 | 0.138074 | 0.159812 |
| 15 | 0.99 | 20000 | 0.334413 | 0.156844 | 0.508743 | 0.132687 | 0.154187 |

Рис. 5 Полученные мной оптимальный портфель, его VaR и $CVaR$.

Видим, что получились результаты, очень близкие к тому, что получили авторы статьи [1] в Таблицах 5 и 6.

6.2. Реализация методов множителей Лагранжа в задаче P3

Требуется решить выпуклую задачу P3 на минимум:

$$x^T V x \rightarrow \min$$

при условии ограничений

$$\begin{cases} x_i \geq 0, i = 1, 2, 3 \\ x_1 + x_2 + x_3 = 1 \\ x_1 m_1 + x_2 m_2 + x_3 m_3 = R \end{cases}$$

Для этого применяем метод множителей Лагранжа, получаем единственную допустимую точку: [0.45423611], [0.11471805], [0.4310484], как и в Таблице 3. Метод множителей Лагранжа приведен ниже, вычисления по полученным формулам проведены на языке Python3 в файле ProblemP3Kurosoyaya.ipynb.

Реализовываем метод множителей Лагранжа для нашей задачи.

$$\Lambda(x, y, z) = \lambda_0 (\bar{x}^T A \bar{x}) - \sum_{i=1}^3 \mu_i x_i - \lambda(x + y + z - 1) - \eta(m_1 x + m_2 y + m_3 z - R)$$

Пишем необходимое условие минимума (достаточность проверять не нужно в силу теоремы Каруша-Куна-Таккера для выпуклой задачи, ведь для нашей задачи очевидно выполнено условие Слейтера):

$$\begin{cases} \Lambda'_x = \lambda_0(2a_{11}x + (a_{12} + a_{21})y + (a_{13} + a_{31})z) - \mu_1 - \lambda - m_1\eta \\ \Lambda'_y = \lambda_0((a_{12} + a_{21})x + 2a_{22}y + (a_{13} + a_{31})z) - \mu_2 - \lambda - m_2\eta \\ \Lambda'_z = \lambda_0((a_{13} + a_{31})x + (a_{23} + a_{32})y + 2a_{33}z) - \mu_3 - \lambda - m_3\eta \\ x + y + z = 1 \\ m_1 x + m_2 y + m_3 z = R \\ \bar{\mu} \bar{x} = 0 \\ \lambda_0, \mu_1, \mu_2, \mu_3, \lambda, \eta \geq 0 \end{cases}$$

Если $\lambda_0 = 0$, то $\begin{cases} \bar{\mu} = -\bar{\lambda} - \eta \bar{m} \\ x + y + z = 1 \\ \bar{\mu} \bar{x} = 0 \\ \bar{m} \bar{x} = R \end{cases}$, откуда разбирая все случаи

$\{x = y = z = 0\}, \{x = 0, y, z \neq 0\}, \{y = 0, x, z \neq 0\}, \{z = 0, x, y \neq 0\}, \{x = y = 0, z \neq 0\}, \{x = z = 0, y \neq 0\}, \{y = z = 0, x \neq 0\}$, получаем, что первый случай противоречит тому, что $x + y + z = 1$, последние три дают несовместную систему типа $\begin{cases} x_i = 1 \\ m_i x_i = R \end{cases}$, а остальные три дают три точки, находимые из системы $\begin{cases} y + z = 1, x = 0 \\ m_1 x + m_2 y = R \end{cases}$, то есть $x = 0, y = \frac{m_3 - R}{m_3 - m_2}, z = \frac{R - m_2}{m_3 - m_2}$; при последующей проверке оказывается, что эти точки не доставляют минимума.

Если $\lambda_0 \neq 0$, то кладем $\lambda_0 = 1$ и, пользуясь симметричностью матрицы A и обозначив за B обратную матрицу, получаем систему

$$\begin{cases} 2A\bar{x} = \bar{\mu} + \bar{\lambda} + \eta \bar{m} \\ x + y + z = 1 \\ \bar{\mu} \bar{x} = 0 \\ \bar{m} \bar{x} = R \end{cases} \Leftrightarrow \begin{cases} \bar{x} = 0.5B(\bar{\mu} + \bar{\lambda} + \eta \bar{m}) \\ x + y + z = 1 \\ \bar{\mu} \bar{x} = 0 \\ \bar{m} \bar{x} = R \end{cases}$$

$$\text{Отсюда} \begin{cases} x = 0.5(\sum_j b_{1j}\mu_j + \lambda \sum_j b_{1j} + \eta \sum_j b_{1j}m_j) \\ y = 0.5(\sum_j b_{2j}\mu_j + \lambda \sum_j b_{2j} + \eta \sum_j b_{2j}m_j) \\ z = 0.5(\sum_j b_{3j}\mu_j + \lambda \sum_j b_{3j} + \eta \sum_j b_{3j}m_j) \\ x + y + z = 1 \\ \bar{m}\bar{x} = R \\ \bar{\mu}\bar{x} = 0 \end{cases}$$

Из последнего получаем, что $\bar{\mu} = 0$, поскольку мы хотим все акции включить в портфель.

$$\begin{cases} x + y + z = 1 \\ \bar{m}\bar{x} = R \end{cases} \Rightarrow \begin{cases} \sum_{i,j} b_{ij}\mu_j + \lambda \sum_{i,j} b_{ij} + \eta \sum_{i,j} b_{ij}m_j = 2 \\ \sum_{i,j} b_{ij}\mu_j m_i + \lambda \sum_{i,j} b_{ij}m_i + \eta \sum_{i,j} b_{ij}m_i m_j = 2R \end{cases}$$

Обозначая $c_{11} = \sum_{i,j} b_{ij}$, $c_{12} = c_{21} = \sum_{i,j} b_{ij}m_i$, $c_{22} = \sum_{i,j} b_{ij}m_i m_j$, получаем систему

$$\begin{cases} c_{11}\lambda + c_{12}\eta = 2 \\ c_{21}\lambda + c_{22}\eta = 2R \end{cases}$$

$$\det = c_{11}c_{22} - c_{12}c_{21}$$

$$\det_\lambda = 2(c_{22} - c_{12}R)$$

$$\det_\eta = 2(c_{11}R - c_{21})$$

$$\lambda = \frac{-2(c_{12}R - c_{22})}{c_{11}c_{22} - c_{12}c_{21}}$$

$$\eta = \frac{2(c_{11}R - c_{21})}{c_{11}c_{22} - c_{12}c_{21}}$$

```

In [2]: m=np.array([0.010111,0.0043532,0.0137058]).reshape(3,1)
m1=m[0]
m2=m[1]
m3=m[2]
R=0.011
a=np.array([0.00324625,0.00022983,0.00420395,0.00022983,0.00049937,0.00019247,0.00420395,0.00019247,0.00764897]).res
b=np.linalg.inv(a)
b11=b[0][0]
b12=b[0][1]
b13=b[0][2]
b21=b[1][0]
b22=b[1][1]
b23=b[1][2]
b31=b[2][0]
b32=b[2][1]
b33=b[2][2]

In [5]: #real
c11=b.sum()
c12=b.dot(m).sum()
c21=c12
c22=m.T.dot(b.dot(m))[0]
c13=2
c23=2*R

det=c11*c22-c12*c21
det_lambda=-(c12*c23-c22*c13)
det_eta=(c11*c23-c21*c13)
lambda=det_lambda/det
eta=det_eta/det

u1=0.5*b.dot(lambda*np.ones(3).reshape(3,1)+eta*m)
print(np.array(u1).reshape(1,-1))

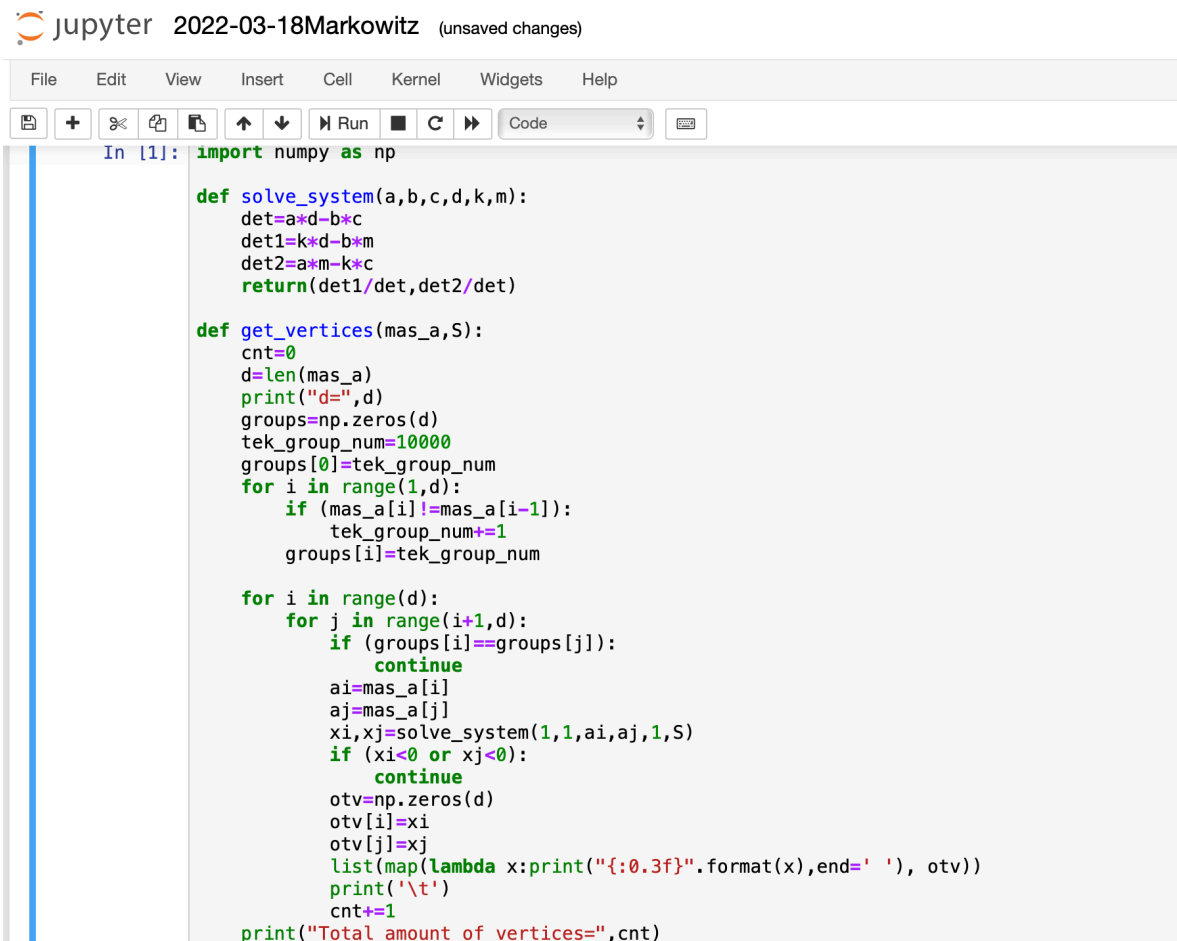
[[0.45423611 0.11471805 0.43104584]]

```

Рис. 6 Вычисление оптимального портфеля по формулам из метода множителей Лагранжа в файле ProblemP3Kurosovaya.ipynb.

Замечание

Отметим, что если в задаче Марковитца мы хотим минимизировать дисперсию (а не максимизировать, как в нашем случае), то способ решения задачи кардинально меняется. А именно, в случае максимизации используется теорема о том, что максимум выпуклой функции на выпуклом множестве достигается в крайней точке. Наше множество является многогранником, крайние точки многогранника — это его вершины, поэтому требуется найти все вершины многогранника. Вершинами будут точки, у которых только две координаты, с номерами i и j будут ненулевые, причем если для этих i и j выполнено $m_i = m_j$, то соответствующая точка вершиной не будет. После нахождения всех вершин многогранника выбирается вершина, доставляющая максимальное значение дисперсии.



```
In [1]: import numpy as np

def solve_system(a,b,c,d,k,m):
    det=a*d-b*c
    det1=k*d-b*m
    det2=a*m-k*c
    return(det1/det,det2/det)

def get_vertices(mas_a,S):
    cnt=0
    d=len(mas_a)
    print("d=",d)
    groups=np.zeros(d)
    tek_group_num=10000
    groups[0]=tek_group_num
    for i in range(1,d):
        if (mas_a[i]!=mas_a[i-1]):
            tek_group_num+=1
            groups[i]=tek_group_num

    for i in range(d):
        for j in range(i+1,d):
            if (groups[i]==groups[j]):
                continue
            ai=mas_a[i]
            aj=mas_a[j]
            xi,xj=solve_system(1,1,ai,aj,1,S)
            if (xi<0 or xj<0):
                continue
            otv=np.zeros(d)
            otv[i]=xi
            otv[j]=xj
            list(map(lambda x:print("{:0.3f}".format(x),end=' '), otv))
            print('\t')
            cnt+=1
    print("Total amount of vertices=",cnt)
```

Рис. 7 Программа для вычисления вершин многогранника для задачи Марковитца в файле 2022-03-18.ipynb.

```
mas_a=[1,1,2,3]
S=5/3
get_vertices(mas_a,S)

d= 4
0.333 0.000 0.667 0.000
0.667 0.000 0.000 0.333
0.000 0.333 0.667 0.000
0.000 0.667 0.000 0.333
Total amount of vertices= 4
```

Рис. 8 Пример работы программы по нахождению вершин

7. Применение к хеджированию

В качестве дальнейшей иллюстрации нашего подхода, сейчас мы рассмотрим пример, в котором хеджируется портфель Nikkei. Эта задача была поставлена Маузером и Розеном (1999). Они рассматривали два варианта хеджирования: параметрический и симуляционный. В каждом случае лучшее хеджирование вычисляется минимизацией VaR вдоль каждой координаты, то есть при фиксации всех позиций портфеля, кроме одной. Здесь мы сначала показываем, что если применять такую же процедуру для минимизации $CVaR$, то получим значения очень близкие к тем, которые получаются при минимизации VaR . Далее мы показываем, что минимизация $CVaR$ имеет преимущество в том, что она применима и при минимизации не вдоль каждой координаты. Позиции нескольких, или даже многих, инструментов могут оптимизироваться одновременно в более широком случае хеджирования.

Как и при применении нашего подхода к задаче оптимизации портфеля в предыдущей секции, вычисления сводятся к задаче линейного программирования способом, описанным после (16) путем добавления дополнительных величин для каждого сценария. Это может давать преимущество для хеджирующих стратегий, которым требуется одновременная оптимизация позиций по большому числу инструментов (более 1000), но тут мы показываем, что для хеджирования относительно небольшого количества инструментов, техники негладкой оптимизации могут сравниться с линейным программированием. При применении таких техник нет нужды вводить дополнительные переменные, поэтому размерность задачи не меняется вне зависимости от того, сколько сценариев мы рассматриваем.

В Таблице 7 показан портфель, который реализует спред бабочки на индекс Nikkei на 1 июля 1997. В добавок к обыкновенным акциям Komatsu и Mitsubishi, портфель содержит некоторое количество опционов пут и колл на эти акции. Этот портфель существенно использует опционы для достижения желаемого вида пэйоффа.

| Instrument | Type | Days to maturity | Strike price (10 ³ JPY) | Position (10 ³) | Value (10 ³ JPY) |
|-----------------------|--------|------------------|------------------------------------|-----------------------------|-----------------------------|
| Mitsubishi EC 6mo 860 | Call | 184 | 860 | 11.5 | 563340 |
| Mitsubishi Corp | Equity | n/a | n/a | 2.0 | 1720000 |
| Mitsubishi Cjul29 800 | Call | 7 | 800 | -16.0 | -967280 |
| Mitsubishi Csep30 836 | Call | 70 | 836 | 8.0 | 382070 |
| Mitsubishi Psep30 800 | Put | 70 | 800 | 40.0 | 2418012 |
| Komatsu Ltd | Equity | n/a | n/a | 2.5 | 2100000 |
| Komatsu Cjul29 900 | Call | 7 | 900 | -28.0 | -11593 |
| Komatsu Cjun2 670 | Call | 316 | 670 | 22.5 | 5150461 |
| Komatsu Cjun2 760 | Call | 316 | 760 | 7.5 | 1020110 |
| Komatsu Paug31 760 | Put | 40 | 760 | -10.0 | -68919 |
| Komatsu Paug31 830 | Put | 40 | 830 | 10.0 | 187167 |

Таблица 7. Состав портфеля Nikkei, заимствованного из работы Mauser and Rosen (1999).

На Графике 1, заимствованном у Mauser and Rosen (1999), показано распределение однодневных убытков на множестве из 1000 сценариев Монте-Карло. Из графика видно, что распределение убытков не похоже на нормальное. Следовательно, подходы минимизации $CVaR$ и минимизации дисперсии могут в этом случае дать различающиеся оптимальные решения (см. Предложение 1 на стр. 9).

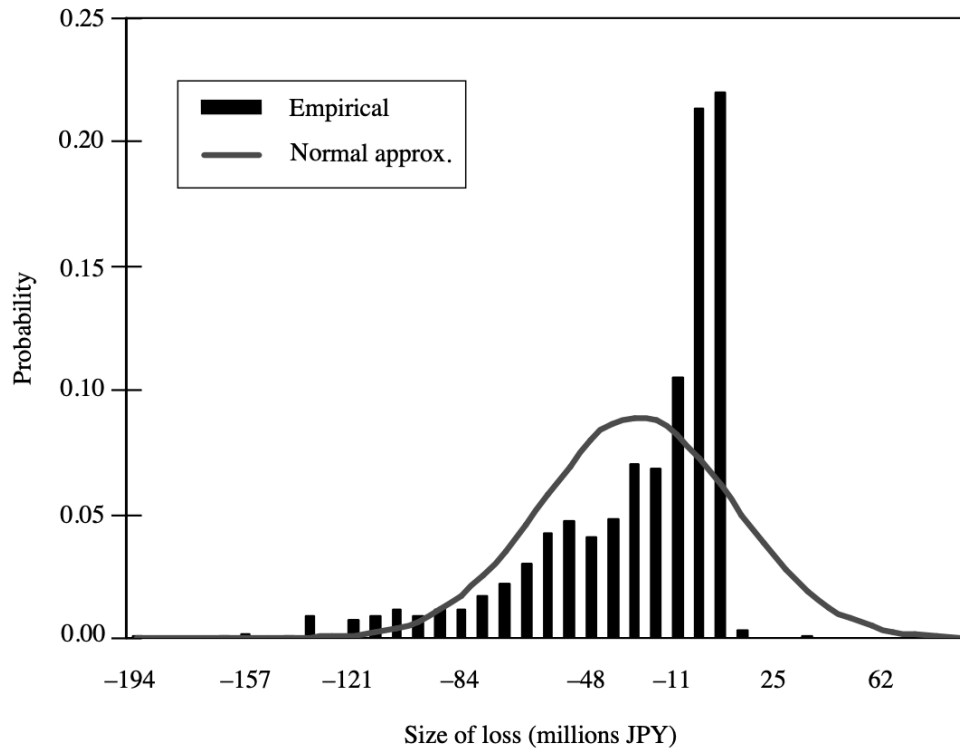


График 1. Эмпирическое распределение однодневных убытков портфеля Nikkei и его приближение нормальной величиной.

Пусть x — вектор позиций по портфелю из 11 инструментов, который мы хотим найти, а z — это вектор начальных позиций, см. пятую колонку в Таблице 7. Эти вектора лежат в \mathbb{R}^{11} . При хеджировании нас интересовало, конечно, только изменение некоторых компонент вектора z , но мы хотели протестировать различные комбинации. Об этом можно думать как о выборе множества индексов J среди $\{1, \dots, 11\}$, обозначающих те инструменты, доли которых мы будем менять. В случае одноинструментного хеджирования, например, мы брали в качестве J набор из одного индекса и последовательно придавали этому индексу все возможные значения.

Выбрав конкретное J , в случае если это J содержит более одного индекса, мы налагали на координаты x_j вектора x следующие ограничения:

$$-|z_j| \leq x_j \leq |z_j|, j \in J \quad (20)$$

$$x_j = z_j, j \notin J \quad (21)$$

$$\text{и полагаем } X = \{x: x \text{ удовлетворяет (20) и (21)}\} \quad (22)$$

Ограничение (21) можно, конечно, использовать для того, чтобы исключить переменные x_j для $j \notin J$, как мы и делаем на практике, но такая формулировка упрощает обозначения и способствует наглядному сравнению различных выборов J . Модуль в (20) возникает из-за того, что z_j могут быть отрицательными, что соответствует короткой позиции по инструменту.

Пусть m — это вектор изначальных цен за единицу инструмента, а y — это случайный вектор цен на следующий день. Следовательно, потери равны начальной цене всего портфеля минус его цене на следующий день:

$$f(x, y) = x^T m - x^T y = x^T (m - y) \quad (23)$$

Соответствующая функция для метода минимизации $CVaR$ выглядит так:

$$F_\beta(x, \alpha) = \alpha + \frac{1}{1-\beta} \int_{y \in \mathbb{R}^{11}} [x^T(m - y) - \alpha]^+ p(y) dy \quad (24)$$

В соответствии с Теоремой 2, нам нужно минимизировать $F_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$. Это минимизация выпуклой функции на выпуклом множестве.

Для аппроксимации интеграла мы генерируем векторы y_1, \dots, y_q и заменяем $F_\beta(x, \alpha)$ на:

$$\widetilde{F}_\beta(x, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^q [x^T(m - y_k) - \alpha]^+ \quad (25)$$

Это выражение опять выпукло по (x, α) и, более того, линейно по каждой координате. Переходя к задаче минимизации $\widetilde{F}_\beta(x, \alpha)$ на множестве $X \times \mathbb{R}$, мы могли бы свести вычисления к задаче линейного программирования, но вместо этого будем решать задачу методом негладкой оптимизации. Для этого мы будем работать с субградиентом множества, соответствующего \widetilde{F}_β в точке (x, α) , который состоит из всех точек множества $\mathbb{R}^{11} \times \mathbb{R}$ вида:

$$(0, 1) + \frac{1}{q(1-\beta)} \sum_{k=1}^q \lambda_k(m - y_k, -1) \quad c \begin{cases} \lambda_k = 1, \text{ если } x^T(m - y_k) - \alpha > 0 \\ \lambda_k \in [0, 1], \text{ если } x^T(m - y_k) - \alpha = 0 \\ \lambda_k = 0, \text{ если } x^T(m - y_k) - \alpha < 0 \end{cases} \quad (26)$$

Мы использовали алгоритм, придуманный Урясьевым в 1991 для негладких задач (см. [2]), взяли $\beta = 0.95$ и получили $\beta\text{-VaR}=657816$ и $\beta\text{-CVaR}=2022060$.

Напомним, в чем заключается метод Урясьева для оптимизации негладких выпуклых функций. Пусть мы хотим найти минимум негладкой выпуклой функции $f(x)$ по $x \in \mathbb{R}^n$. Предлагается искать решение по следующей рекуррентной формуле:

$$x^{s+1} = x^s - \rho_s H^s g^s; s = 0, 1 \dots$$

Здесь ρ_s — параметр шага (выбирается нами, например, 0.5), g^s — субградиент (то есть любой вектор из субдифференциала функции $f(x)$ в точке x^s), H^s — матрица $n \times n$, выбираемая из условия минимизации функции $\varphi_s(H) := f(x^s - \rho_s H^s g^s)$ по H^s . Задача нахождения оптимального H^s решается также методом субградиентного спуска, то есть делается шаг в сторону антисубградиента функции $\varphi_s(H)$ до тех пор, пока не выполнится неравенство

$$f(x^s - \rho_s H^s g^s) < f(x^s - \rho_s H^{s,old} g^s).$$

В файле MetodUryaseva.cpp написана реализация метода Урясьева на языке программирования на C++. Для модельной функции

$$f(x_1, x_2) = 100(x_1^2 - x_2) + (x_1 - 1)^2$$

проверено, что теоретический минимум совпадает с численным. А именно, написанный мной алгоритм численно нашел правильную теоретическую точку минимума (1,1) за 572 шага.

Теперь вернемся к применению метода Урясьева для задачи хеджирования. Результаты оптимизации по одному инструменту, когда мы последовательно брали $J = \{1\}$, $J = \{2\}$, ..., $J = \{11\}$, представлены в Таблице 8.

| Instrument | Best hedge | VaR | CVaR |
|-----------------------|------------|----------|---------|
| Mitsubishi EC 6mo 860 | 7337.53 | -205927 | 1183040 |
| Mitsubishi Corp | -926.073 | -1180000 | 551892 |
| Mitsubishi Cjul29 800 | -18978.6 | -1170000 | 553696 |
| Mitsubishi Csep30 836 | 4381.22 | -1150000 | 549022 |
| Mitsubishi Psep30 800 | 43637.1 | -1150000 | 542168 |
| Komatsu Ltd | -196.167 | -1180000 | 551892 |
| Komatsu Cjul29 900 | -124939 | -1200000 | 593078 |
| Komatsu Cjun2 670 | 19964.9 | -1220000 | 385698 |
| Komatsu Cjun2 760 | 4745.20 | -1200000 | 363556 |
| Komatsu Paug31 760 | 31426.3 | -1120000 | 538662 |
| Komatsu Paug31 830 | 19356.3 | -1150000 | 536500 |

Таблица 8. Наилучший хедж, его VaR и $CVaR$ в задаче минимизации $CVaR$ для одного инструмента с $\beta = 0.95$.

Полученные оптимальные хеджи близки к тем, которые были получены Mauser and Rosen (1999) методом минимизации β - VaR . Поскольку J состоит из одного индекса, то в наших тестах вектор x был одномерным. Минимизация по (x, α) была бы двумерной. Алгоритму требуется менее 100 итераций для нахождения 6 верных знаков у ответа.

Закончив с одноинструментным хеджированием, мы попробовали хеджирование с изменением позиций инструментов с 4 по 11 одновременно. Полученные оптимальные хеджи показаны в Таблице 9.

| Instrument | Position in portfolio | Best hedge |
|--------------------|-----------------------|------------|
| Komatsu Cjun2 670 | 22500 | 22500 |
| Komatsu Cjun2 760 | 7500 | -527 |
| Komatsu Paug31 760 | -10000 | -10000 |
| Komatsu Paug31 830 | 10000 | -10000 |

Таблица 9. Начальная позиция и наилучший хедж в задаче минимизации $CVaR$ для четырех инструментов с $\beta = 0.95$.

Оптимизация не изменила позиции по инструментам Komatsu Cjun2 670 и Komatsu Paug31 760, но позиции по Komatsu Cjun2 760 и Komatsu Paug31 830 изменили не только модуль, но и знак. В сравнении с одноинструментным хеджированием, мы видим, что многоинструментное хеджирование существенно уменьшило как VaR , так и $CVaR$. А именно, получилось β - $VaR = -1400000$ и β - $CVaR = 37334.6$, то есть меньше, чем в одномерном случае (β - $VaR = -1200000$ и β - $CVaR = 363556$, см. Строка 9 в Таблице 8).

В отличие от предыдущей части, где мы использовали метод линейного программирования, здесь мы использовали метод негладкой оптимизации, что позволило не увеличивать размерность задачи при росте количества сценариев. Это дает вычислительные преимущества, если параметров у задачи очень много.

Этот пример показывает, что при вычислении рисков предпочтительнее минимизировать $CVaR$ вместо VaR . Приведенные портфели имеют положительное $CVaR$ при отрицательном VaR при одном и том же $\beta = 0.95$. Например, портфель, соответствующий первой строке Таблицы 8, имеет β - $VaR = -205927$ и β - $CVaR =$

1183040. Отрицательные потери — это прибыль. То есть кажется, что такой портфель с вероятностью не меньше 0.95 даст прибыль в размере 205927. Но эта цифра не показывает, насколько плохи могут быть дела в остальных 5% случаях. Таблица с β - $CVaR$ показывает, что в тех случаях, когда прибыль меньше 205927 — то у нас вообще потери в среднем в размере 1183040.

8. Выводы

Мы рассмотрели новый подход к минимизации $CVaR$ и одновременному вычислению VaR для широкого класса задач. Мы показали, что $CVaR$ можно эффективно искать с помощью методов линейного программирования и негладкой оптимизации. Хотя, формально, наш метод оптимизирует только $CVaR$, но наши численные эксперименты показали, что наш метод также минимизирует VaR , потому что $VaR \leq CVaR$.

Мы продемонстрировали наши результаты на двух примерах. Оба этих примера имеют сравнительно небольшие размерности и приведены именно в качестве иллюстрации правильной работы метода.

Виден потенциал для улучшения предложенного подхода. Например, предположение о существовании совместной плотности распределений инструментов можно опустить. Далее, можно сделать оптимизацию при условии ограничений, содержащих VaR . Наконец, можно применять различные методы линейного программирования и негладкой оптимизации и исследовать, какие из них дают более точные и быстрые результаты.

9. Приложение

Все программы на C и Python, написанные в ходе выполнения работы, можно найти в репозитории по ссылке <https://github.com/tokaevaAA/kursovaya5>.

10. Список литературы

- 1) Rockafellar, R. and Uryasev, S. (2000) Optimization of Conditional Value at Risk, Journal of Risk, 2, 21-41.
- 2) Uryasev, S. (1991) New variable-metric algorithms for nondifferentiable optimization problems, Journal of optimization theory and applications, 71-2, 359-388.
- 3) Rockafellar, R. and Uryasev, S. (2002) Conditional Value at Risk for General Loss Distributions, Journal of Banking and Finance, 26, 1443-1471.
- 4) S. Uryasev, S. Sarykalin, G. Serraino (2008), Value-at-Risk vs Conditional Value-at-Risk in Risk Management and Optimization, Conference paper, DOI: 10.1287/educ.1080.0052.