ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 330.45

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МАТЕМАТИЧЕСКИХ И ПРОГРАММНЫХ ИНСТРУМЕНТОВ В ОПТИМИЗАЦИИ ЭКОНОМИЧЕСКИХ ЗАДАЧ

Алипанова Виктория Павловна,
Руководитель Центр информационного сопровождения образовательной деятельности и качества образования, и.о. экономиста, преподаватель первой квалификационной категории Самарского колледжа цифровой экономики и предпринимательства «МИР», 443030 Россия, г. Самара, ул. Г. Аксакова, 21.
E-mail: alipanova_viktoriya@mail.ru

Аннотация

В статье предложен компактный прикладной инструмент для решения задачи распределения фонда материальной помощи обучающихся, основанный на целочисленной математической модели и её реализации на Python с пользовательским интерфейсом в Microsoft Excel. Инструмент выполняет автоматизированный перебор допустимых вариантов в заданных диапазонах параметров и выводит результаты на лист таблицы для последующего анализа. Приведена формальная постановка задачи, оценена вычислительная трудоёмкость алгоритма и даны количественные результаты на реальных параметрах фонда.

Ключевые слова: задачи оптимизации, распределение фонда, целочисленная модель, Python, Microsoft Excel; VBA-макрос; поддержка принятия решений.

PRACTICAL APPLICATION OF MATHEMATICAL AND SOFTWARE TOOLS TO THE OPTIMIZATION OF ECONOMIC PROBLEMS

Alipanova Victoria Pavlovna,
Head of the Center for information support of educational activities and quality of education, acting economist, first-Category Lecturer, Samara College of digital economy and entrepreneurship «MIR», 443030, Russia, Samara, G. Aksakova St., 21.
E-mail: alipanova_viktoriya@mail.ru

ABSTRACT

The article proposes a compact applied tool for solving the problem of allocating a student financial support fund, based on an integer mathematical model and its implementation in Python with a Microsoft Excel user interface. The tool performs an automated enumeration of feasible options within the specified parameter ranges and outputs the results to a worksheet for subsequent analysis. A formal problem statement is given, the computational complexity of the algorithm is estimated, and quantitative results on real fund parameters are reported.

Keywords: optimization problems; fund allocation; integer model; Python; Microsoft Excel; VBA macro; decision support.

Введение

В повседневной работе экономиста и аналитика большинство расчётов по-прежнему выполняется в табличных редакторах, что при задачах распределения ресурсов с несколькими ограничениями приводит к громоздким файлам и ручным переборам, сопряжённым с риском ошибок. Современные исследования подтверждают, что внедрение формализованных математических моделей и средств автоматизации существенно повышает эффективность управленческих расчётов и обоснованность принимаемых решений [1; 2]. Вместе с тем методы исследования операций и целочисленной оптимизации сравнительно редко доводятся до удобных прикладных инструментов в среде, привычной для специалиста, прежде всего в Microsoft Excel, что и формирует устойчивый разрыв между теоретическими методами и реальной практикой [4].

Автор статьи столкнулся с этим разрывом при решении прикладной задачи распределения фонда материальной помощи обучающихся: при фиксированном бюджете и жёстких ограничениях по числу получателей и размерам выплат ручной поиск допустимых комбинаций параметров оказался малоэффективным.

Особенностью работы является то, что предложенное решение разработано и апробировано практикующим экономистом образовательной организации в ответ на конкретный запрос повседневной деятельности, что повышает его прикладную значимость для учреждений со сходными нормативными ограничениями [3]. Практическая новизна состоит в том, что классические методы целочисленной оптимизации переносятся в инструментальную среду, не требующую от конечного пользователя навыков программирования.

Цель исследования

Разработать и описать компактный инструмент для решения экономических задач оптимизационного характера, основанный на целочисленной математической модели и программной реализации на Python с интерфейсом в среде Excel, на примере задачи распределения фонда материальной помощи обучающихся.

Материалы и методы исследования

Объектом исследования выступает прикладная экономическая задача распределения ограниченного фонда между получателями при заданных нормативных ограничениях по числу участников и размеру выплат. Исходные параметры, интервалы и ограничения сформированы на основе реальных условий работы экономиста образовательной организации.

В исследовании применяются методы анализа и синтеза для формулировки экономической постановки задачи, методы формализация и математическое моделирование применялись для построения целочисленной модели, а также вычислительный эксперимент на основе перебора допустимых решений в заданных диапазонах [5]. Построенная модель реализована в виде программного инструмента на Python с пользовательским интерфейсом в Microsoft Excel.

Формальная постановка задачи. Пусть F – общий объём фонда материальной помощи, доступный для распределения между двумя семестрами; n_1, n_2 – целое число получателей в первом и втором семестрах; a_1, a_2 – размер единичной выплаты в первом и втором семестрах (руб.). Тогда задача формулируется как поиск множества допустимых

целочисленных решений (n_1, a_1, n_2, a_2), удовлетворяющих балансовому равенству и двусторонним ограничениям:

$n_1 \cdot a_1 + n_2 \cdot a_2 = F$,

$n_i\_min \le n_i \le n_i\_max$, $a_i\_min \le a_i \le a_i\_max$, $i \in \{1, 2\}$,

$n_i \in \mathbb{Z}$, a_i кратно шагу step.

В отличие от классической постановки задачи линейного программирования здесь не задаётся единый скалярный критерий оптимальности: основной задачей инструмента является построение полного множества допустимых решений с последующим многокритериальным анализом (по равномерности распределения, общей сумме, разнице в числе получателей и размерах выплат). Такая постановка соответствует логике принятия управленческих решений в условиях, когда формальный «оптимум» не имеет однозначной экономической интерпретации, а решение выбирается человеком по нескольким критериям одновременно [3].

Оценка вычислительной трудоёмкости. При диапазонах $n_i \in [40; 60]$ (21 значение) и $a_i \in [4\,000; 5\,500]$ с шагом 10 руб. (151 значение) общее число просматриваемых комбинаций по схеме «перебор по n_1, n_2, a_1 с вычислением a_2 из балансового равенства» составляет $21 \times 21 \times 151 \approx 6{,}6 \cdot 10^4$ итераций, что отрабатывается на современном персональном компьютере за единицы секунд. Полный перебор всех четырёх переменных без оптимизации схемы потребовал бы около $21 \times 21 \times 151 \times 151 \approx 10^7$ итераций, что также допустимо в данной размерности задачи. При расширении модели (введение нескольких категорий получателей, дополнительных периодов или вероятностных ограничений) трудоёмкость растёт мультипликативно, поэтому при числе переменных более 6–8 целесообразен переход к специализированным решателям (PuLP, Pyomo, SciPy.optimize) [2].

Результаты и их обсуждение

Разработанный программный инструмент сочетает перебор целочисленных вариантов в среде Python и удобный интерфейс для анализа результатов в Microsoft Excel. Пользователь задаёт параметры задачи на листе Excel «Параметры», после чего автоматически формируется множество допустимых схем распределения фонда и отображается в табличном виде на листе «Решения» (см. рис. 1). Такое разделение ролей объединяет преимущества обоих подходов: высокую скорость вычислений Python и наглядность Excel при последующем анализе.

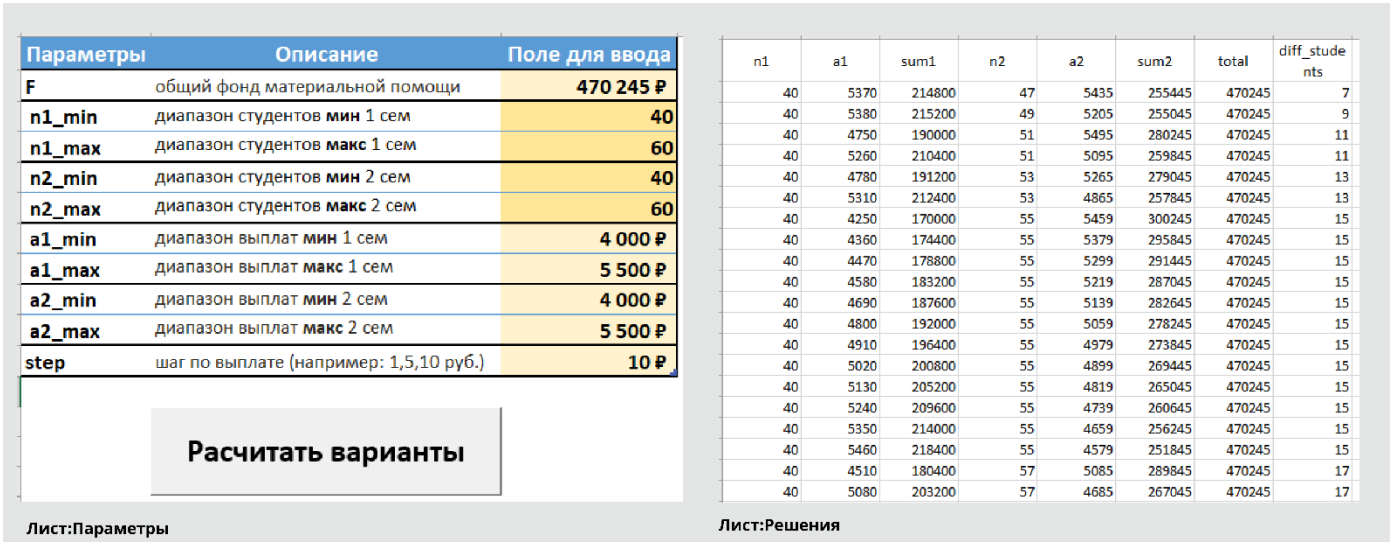


Рисунок 1 – Интерфейс инструмента: лист «Параметры» с полями ввода и кнопкой запуска расчёта; лист «Решения» с таблицей найденных вариантов

На листе «Параметры» задаются общий объём фонда, минимальное и максимальное количество обучающихся в первом и втором семестрах, диапазоны возможных размеров выплат, а также шаг изменения выплат. Здесь же размещена кнопка «Рассчитать варианты», по нажатию которой запускается автоматический расчёт; по его завершении пользователь получает сообщение с указанием количества найденных решений, что позволяет оценить, насколько жёсткими или, наоборот, свободными являются заданные ограничения.

Для связи вычислительной части с пользовательским интерфейсом разработан макрос «FindScholarshipSolutions» на языке VBA (см. рис. 2). Макрос считывает параметры задачи с листа «Параметры», подготавливает лист «Решения» (очищает старые данные и формирует заголовки столбцов). Каждая строка таблицы соответствует отдельному варианту распределения и содержит значения n1, a1, sum1, n2, a2, sum2, total, diff_students, diff_amount, что позволяет наглядно сравнивать схемы между собой.

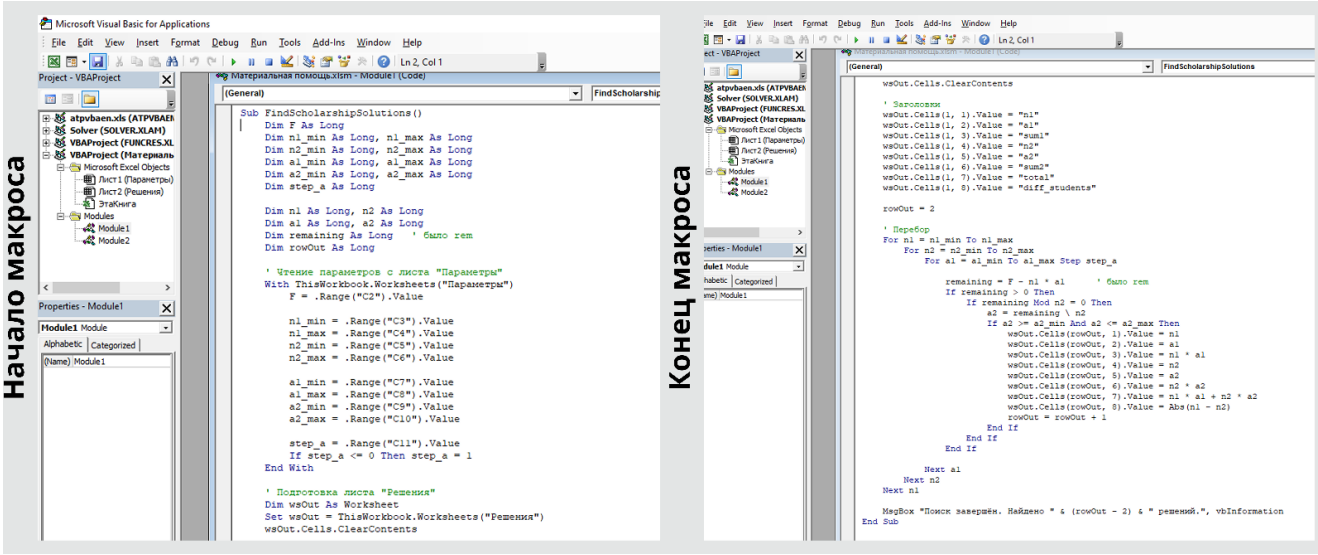
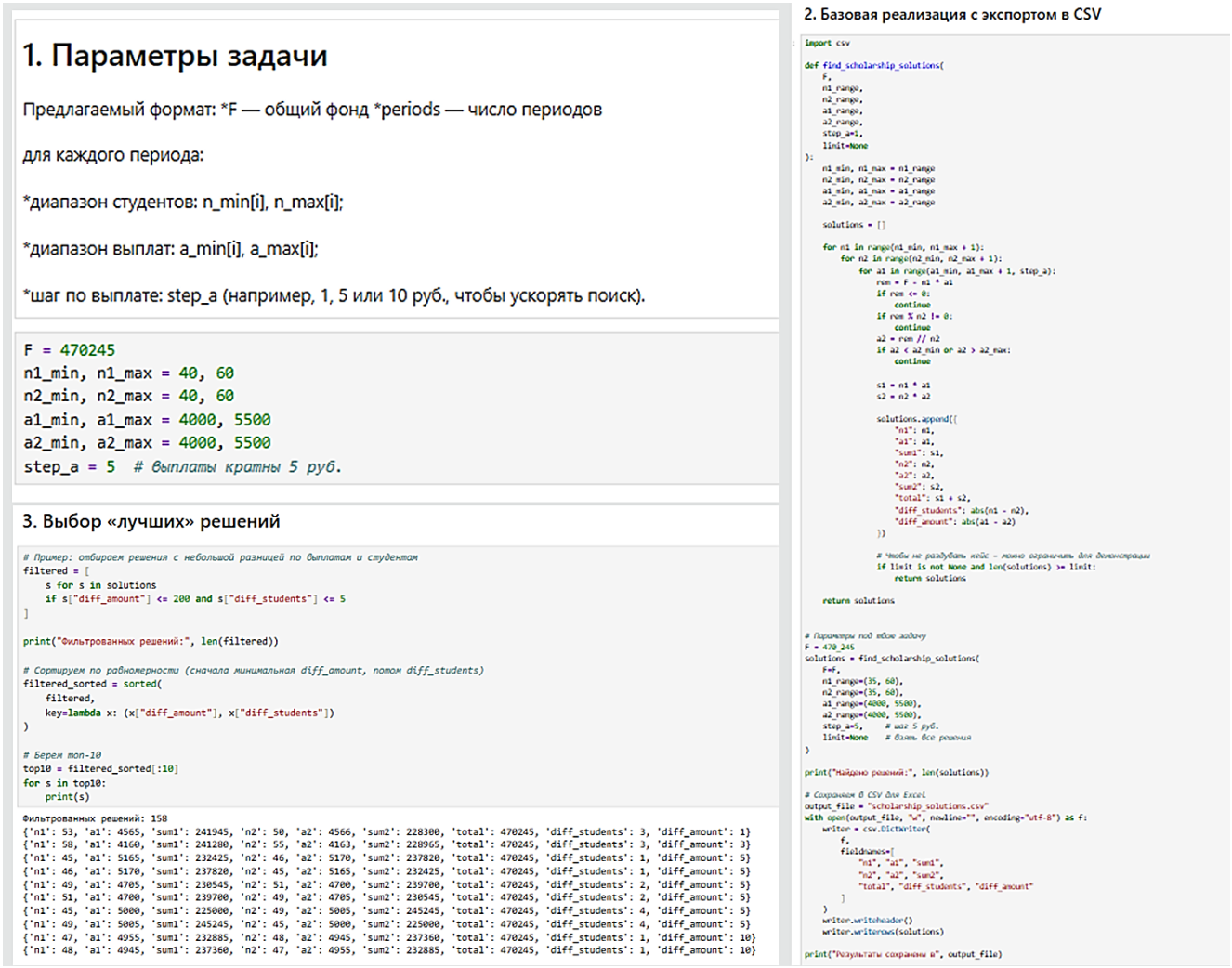


Рисунок 2 – Окно редактора VBA с кодом макроса «FindScholarshipSolutions»

Вычислительная логика реализована на языке Python в виде функции «find_scholarship_solutions», запускаемой в Jupyter Notebook (см. рис. 3). Функция последовательно перебирает все допустимые комбинации значений в заданных диапазонах с учётом шага и отбирает те из них, которые удовлетворяют балансовому равенству. Для каждой найденной комбинации дополнительно вычисляются суммарные выплаты по семестрам, общая сумма, разница в числе обучающихся и разница в размере выплат, что позволяет сразу перейти к анализу равномерности распределения [5].

Сформированный список решений сохраняется в CSV-файл с фиксированным набором полей, совместимых с табличным представлением в Excel. На этапе вычислений предусмотрен также пример автоматического отбора «лучших» вариантов: по заданным порогам на diff_students и diff_amount из полного множества выделяется подмножество более сбалансированных решений, которое сортируется по степени равномерности. Критерии отбора можно изменить в коде без перестройки основной структуры алгоритма, что делает инструмент гибким для других прикладных задач распределения ресурсов.



1. Параметры задачи

Предлагаемый формат: *F — общий фонд *periods — число периодов

для каждого периода:

*диапазон студентов: n_min[i], n_max[i];

*диапазон выплат: a_min[i], a_max[i];

*шаг по выплате: step_a (например, 1, 5 или 10 руб., чтобы ускорять поиск).

```
F = 470245
n1_min, n1_max = 40, 60
n2_min, n2_max = 40, 60
a1_min, a1_max = 4000, 5500
a2_min, a2_max = 4000, 5500
step_a = 5  # выплаты кратны 5 руб.
```

3. Выбор «лучших» решений

```
# Пример: отбираем решения с небольшой разницей по выплатам и студентам
filtered = [
    s for s in solutions
    if s["diff_amount"] <= 200 and s["diff_students"] <= 5
]

print("Фильтрованных решений:", len(filtered))

# Сортируем по равномерности (сначала минимальная diff_amount, потом diff_students)
filtered_sorted = sorted(
    filtered,
    key=lambda x: (x["diff_amount"], x["diff_students"])
)

# Берем топ-10
top10 = filtered_sorted[:10]
for s in top10:
    print(s)
```

```
Фильтрованных решений: 158
{'n1': 53, 'a1': 4565, 'sum1': 241945, 'n2': 50, 'a2': 4566, 'sum2': 228300, 'total': 470245, 'diff_students': 3, 'diff_amount': 1}
{'n1': 58, 'a1': 4160, 'sum1': 241280, 'n2': 55, 'a2': 4163, 'sum2': 228965, 'total': 470245, 'diff_students': 3, 'diff_amount': 3}
{'n1': 45, 'a1': 5165, 'sum1': 232425, 'n2': 46, 'a2': 5170, 'sum2': 237820, 'total': 470245, 'diff_students': 1, 'diff_amount': 5}
{'n1': 46, 'a1': 5170, 'sum1': 237820, 'n2': 45, 'a2': 5165, 'sum2': 232425, 'total': 470245, 'diff_students': 1, 'diff_amount': 5}
{'n1': 49, 'a1': 4705, 'sum1': 230545, 'n2': 51, 'a2': 4700, 'sum2': 239700, 'total': 470245, 'diff_students': 2, 'diff_amount': 5}
{'n1': 51, 'a1': 4700, 'sum1': 239700, 'n2': 49, 'a2': 4705, 'sum2': 230545, 'total': 470245, 'diff_students': 2, 'diff_amount': 5}
{'n1': 45, 'a1': 5000, 'sum1': 225000, 'n2': 49, 'a2': 5005, 'sum2': 245245, 'total': 470245, 'diff_students': 4, 'diff_amount': 5}
{'n1': 49, 'a1': 5005, 'sum1': 245245, 'n2': 45, 'a2': 5000, 'sum2': 225000, 'total': 470245, 'diff_students': 4, 'diff_amount': 5}
{'n1': 47, 'a1': 4955, 'sum1': 232885, 'n2': 48, 'a2': 4945, 'sum2': 237360, 'total': 470245, 'diff_students': 1, 'diff_amount': 10}
{'n1': 48, 'a1': 4945, 'sum1': 237360, 'n2': 47, 'a2': 4955, 'sum2': 232885, 'total': 470245, 'diff_students': 1, 'diff_amount': 10}
```

2. Базовая реализация с экспортом в CSV

```
import csv

def find_scholarship_solutions(
    F,
    n1_range,
    n2_range,
    a1_range,
    a2_range,
    step_a=5,
    limit=None
):
    n1_min, n1_max = n1_range
    n2_min, n2_max = n2_range
    a1_min, a1_max = a1_range
    a2_min, a2_max = a2_range

    solutions = []

    for n1 in range(n1_min, n1_max + 1):
        for n2 in range(n2_min, n2_max + 1):
            for a1 in range(a1_min, a1_max + 1, step_a):
                rem = F - n1 * a1
                if rem <= 0:
                    continue
                if rem % n2 != 0:
                    continue
                a2 = rem // n2
                if a2 < a2_min or a2 > a2_max:
                    continue

                s1 = n1 * a1
                s2 = n2 * a2

                solutions.append({
                    "n1": n1,
                    "a1": a1,
                    "sum1": s1,
                    "n2": n2,
                    "a2": a2,
                    "sum2": s2,
                    "total": s1 + s2,
                    "diff_students": abs(n1 - n2),
                    "diff_amount": abs(a1 - a2)
                })

                # Чтобы не раздувать кейс – можно ограничить для демонстрации
                if limit is not None and len(solutions) >= limit:
                    return solutions

    return solutions


# Параметры под вашу задачу
F = 470_245
solutions = find_scholarship_solutions(
    F=F,
    n1_range=(35, 60),
    n2_range=(35, 60),
    a1_range=(4000, 5500),
    a2_range=(4000, 5500),
    step_a=5,     # шаг 5 руб.
    limit=None    # взять все решения
)

print("Найдено решений:", len(solutions))

# Сохраняем в CSV для Excel
output_file = "scholarship_solutions.csv"
with open(output_file, "w", newline="", encoding="utf-8") as f:
    writer = csv.DictWriter(
        f,
        fieldnames=[
            "n1", "a1", "sum1",
            "n2", "a2", "sum2",
            "total", "diff_students", "diff_amount"
        ]
    )
    writer.writeheader()
    writer.writerows(solutions)

print("Результаты сохранены в", output_file)
```

Рисунок 3 – Фрагмент Jupyter Notebook с заданием параметров задачи, функцией «find_scholarship_solutions» и примером отбора «лучших» решений

Количественные результаты апробации. Инструмент апробирован на реальных параметрах фонда материальной помощи: F = 470 245 руб., $n_i \in [40; 60]$, $a_i \in [4\,000; 5\,500]$ руб. с шагом 10 руб. При указанных условиях из приблизительно $6{,}6 \cdot 10^4$ просматриваемых комбинаций было выделено 542 допустимых решения, удовлетворяющих балансовому равенству и нормативным ограничениям. Время расчёта на персональном компьютере составило менее двух секунд. Для сравнения, ручной подбор удовлетворительного варианта в табличном редакторе занимал ранее не менее одного часа и не давал гарантии полноты охвата возможных схем распределения. Полученный набор решений далее анализируется стандартными средствами Excel: пользователь применяет автофильтр и сортировку по интересующим столбцам, например, по разнице в числе обучающихся или размеру выплат между семестрами. Это даёт возможность быстро выделять варианты с наиболее равномерным распределением и проверять, как меняются решения при корректировке исходных параметров.

Ограничения применимости. В текущей постановке задача рассматривает два периода распределения (семестра), единый размер выплаты внутри каждого периода и одну категорию получателей. Эти допущения соответствуют пилотному кейсу образовательной организации; их снятие предполагается на последующих этапах развития инструмента.

Перспективным направлением является масштабирование инструмента за счёт более полного учёта реальных условий распределения фонда. На следующем этапе возможна

интеграция статистики за предыдущие периоды: фактического количества обучающихся, доли студентов на бюджетной основе, структуры обращений за материальной помощью и объёмов выделяемых средств, что позволит не только задавать диапазоны параметров вручную, но и автоматически формировать их на основе исторических данных, прогнозируя ожидаемое количество получателей и типичные размеры выплат [6]. Дальнейшее развитие модели предполагает также включение нескольких категорий получателей с различными нормативами (по курсам, направлениям подготовки или социальным группам) и введение вероятностных ограничений, отражающих неопределённость в численности обучающихся и объёме фонда. В таком виде система будет представлять собой масштабируемый инструмент поддержки принятия решений для экономистов образовательных организаций, позволяющий оценивать устойчивость найденных схем к изменениям исходных условий [3].

Таким образом, разработанное решение демонстрирует, как методы целочисленной оптимизации и вычислительный эксперимент могут быть «упакованы» в компактный прикладной инструмент, доступный экономисту без специальной подготовки в программировании. Реализация математической модели на Python с интерфейсом в среде Excel существенно снижает трудоёмкость расчётов, сокращает число ошибок ручного перебора и делает процесс распределения ограниченного фонда более прозрачным и обоснованным. Поставленная в работе цель достигается не только на уровне конкретного кейса, но и как иллюстрация более общего подхода к интеграции математических моделей в повседневную практику экономиста.

Список литературы:

1. Артемьева С. И., Кублашвили О. В. Разработка модельных решений по оптимизации бизнес-процессов при внедрении системы менеджмента качества // Омский научный вестник. Серия «Общество. История. Современность». – 2024. – Т. 9, № 3. – С. 171–178.
2. Барганалиева Ж. К., Султанбаева Г. С., Асанова Ж. К., Асанбекова Н. О. Решение задач линейного программирования с помощью библиотеки Pyomo на языке Python // Международный журнал прикладных и фундаментальных исследований. – 2023. – № 12. – С. 29–32. – DOI: 10.17513/mjpfi.13601. – URL: https://applied-research.ru/article/view?id=13601
3. Бурлов В. Г., Грачев М. И. Показатель автоматизации как критерий эффективности реализации управленческих решений в организации // XXI век: итоги прошлого и проблемы настоящего плюс. – 2023. – Т. 12, № 3 (63). – С. 57–65.
4. Грачев М. И. Повышение эффективности работы организации на основе критерия автоматизации // Интеллектуальные системы в производстве. – 2023. – Т. 21, № 3. – С. 144–150. – DOI: 10.22213/2410-9304-2023-3-144-150.
5. Тишкин М. С. Значимость принятия управленческих решений в функционировании современных организаций // Вестник РЭУ им. Г. В. Плеханова. – 2024. – Т. 21, № 2 (134). – С. 161–170.
6. Чеботарева В. А., Алфимцев А. Н. Оптимизация бизнес-процессов с помощью внедрения информационных технологий // Молодежный научный форум. – 2022. – № 9 (177). – С. 38–41.

References:

1. Artemyeva S. I., Kublashvili O. V. Development of model solutions for optimizing business processes during the implementation of a quality management system // Omsk Scientific Bulletin. Series "Society. History. Modernity". - 2024. - Vol. 9, No. 3. - Pp. 171-178.
2. Barganalieva Zh. K., Sultanbaeva G. S., Asanova Zh. K., Asanbekova N. O. Solving linear programming problems using the Pyomo library in Python // International Journal of Applied and Fundamental Research. - 2023. - No. 12. - Pp. 29-32. - DOI: 10.17513/mjpfi.13601. – URL: https://applied-research.ru/article/view?id=13601
3. Burlov V. G., Grachev M. I. Automation indicator as a criterion for the effectiveness of management decisions in an organization // 21st century: results of the past and problems of the present plus. – 2023. – Vol. 12, No. 3 (63). – Pp. 57–65.
4. Grachev M. I. Improving the efficiency of an organization based on the automation criterion // Intelligent systems in production. – 2023. – Vol. 21, No. 3. – Pp. 144–150. – DOI: 10.22213/2410-9304-2023-3-144-150.
5. Tishkin M. S. The importance of making management decisions in the functioning of modern organizations // Bulletin of the Plekhanov Russian University of Economics. G. V. Plekhanov. - 2024. - Vol. 21, No. 2 (134). - P. 161-170.
6. Chebotareva V. A., Alfimtsev A. N. Optimization of business processes through the implementation of information technologies // Youth Scientific Forum. - 2022. - No. 9 (177). - P. 38-41.