КОМПЬЮТЕРНЫЕ НАУКИ И ИНФОРМАТИКА

Научная статья
УДК 004.021:004.94
DOI: 10.17072/1993-0550-2026-2-88-103
https://elibrary.ru/ddcbii

Исследование алгоритмической сложности построения Парето-множества для принятия решения при распределении ресурсов вычислительных систем

Егор Сергеевич Трушкин
Пермский национальный исследовательский политехнический университет, Пермь, Россия
egor.s.trushkin@gmail.com

Аннотация. Современные вычислительные системы функционируют в условиях гетерогенности узлов и переменной нагрузки, что обусловливает необходимость применения многокритериальных методов при распределении вычислительных ресурсов. Одним из таких методов является предсказательный метод распределения ресурсов на основе многокритериальной модели принятия решений, ключевым этапом которого служит построение множества Парето-оптимальных решений. Вместе с тем вычислительная сложность данного этапа и возможности ее снижения ранее не исследовались. Объектом исследования являются алгоритмы построения Парето-множества в задаче предсказательного распределения ресурсов в гетерогенных вычислительных системах. Цель работы состоит в анализе вычислительной сложности базового алгоритма оптимизации и исследовании методов ее снижения применительно к задачам многокритериального распределения ресурсов. Методология: в работе формализована постановка задачи построения Парето-множества, установлена теоретическая оценка сложности базового алгоритма. Рассмотрены и проанализированы три метода снижения вычислительной сложности: предварительная фильтрация альтернатив по ключевому критерию, ранняя остановка при попарном сравнении и инкрементальное построение Парето-множества. Экспериментальная часть: для экспериментального подтверждения теоретических оценок разработана программа моделирования, с помощью которой проведены три серии экспериментов: исследование зависимости от числа узлов n, от числа критериев k и от порогового параметра θ метода фильтрации. Полученные результаты позволяют обоснованно выбирать алгоритм построения Парето-множества в зависимости от масштаба системы, числа критериев и требований к полноте решения, что способствует повышению эффективности планировщика в многокритериальных системах распределения ресурсов реального времени.

Ключевые слова: *вычислительная система; распределение ресурсов; Парето-оптимальность; многокритериальная оптимизация; ранняя остановка; инкрементальный алгоритм; предварительная фильтрация; прогнозирование нагрузки.*

Для цитирования: *Трушкин Е. С.* Исследование алгоритмической сложности построения Парето-множества для принятия решения при распределении ресурсов вычислительных систем //

Вестник Пермского университета. Математика. Механика. Информатика. 2026. № 2(73). С. 88−103. DOI: 10.17072/1993-0550-2026-2-88-103. https://elibrary.ru/ddcbii.

Статья поступила в редакцию 21.04.2026; одобрена после рецензирования 20.05.2026; принята к публикации 10.06.2026.

COMPUTER SCIENCE

Research article

Study of the Algorithmic Complexity of Constructing a Pareto Set for Decision-Making in Allocating Resources of Computing Systems

Egor S. Trushkin

Perm National Research Polytechnic University, Perm, Russia
egor.s.trushkin@gmail.com

Abstract. Modern computing systems operate in conditions of heterogeneity of nodes and variable load, which necessitates the use of multi-criteria methods in the allocation of computing resources. One of these methods is the predictive method of resource allocation based on a multi-criteria decision-making model, the key stage of which is the construction of a set of Pareto-optimal solutions. However, the computational complexity of this stage and the possibility of reducing it have not been previously investigated. The object of the research is algorithms for constructing a Pareto set in the problem of predictive resource allocation in heterogeneous computing systems. The purpose of the work is to analyze the computational complexity of the basic optimization algorithm and to study methods for reducing it in relation to multi-criteria resource allocation tasks. Methodology: the paper formalizes the problem of constructing a Pareto set, establishes a theoretical estimate of the complexity of the basic algorithm. Three methods of reducing computational complexity are considered and analyzed: preliminary filtering of alternatives by a key criterion, early stopping in pairwise comparison, and incremental construction of a Pareto set. Experimental part: to experimentally confirm the theoretical estimates, a simulation program has been developed, with the help of which three series of experiments have been conducted: a study of the dependence on the number of nodes n, on the number of criteria k, and on the threshold parameter θ of the filtration method. The results obtained make it possible to reasonably choose an algorithm for constructing a Pareto set depending on the scale of the system, the number of criteria and the requirements for completeness of the solution, which helps to increase the efficiency of the scheduler in multi-criteria real-time resource allocation systems.

Keywords: *computing systems; resource allocation; Pareto optimality; multi-criteria optimization; early shutdown; incremental algorithm; pre-filtering; load forecasting.*

For citation: Trushkin, E. S. (2026), "Study of the Algorithmic Complexity of Constructing a Pareto Set for Decision-Making in Allocating Resources of Computing Systems", *Bulletin of Perm University. Mathematics. Mechanics. Computer Science*, no 2(73), pp. 88−103, DOI: 10.17072/1993-0550-2026-2-88-103, https://elibrary.ru/ddcbii.

The article was submitted 21.04.2026; approved after reviewing 20.05.2026; accepted for publication 10.06.2026.

Введение

Задача распределения ресурсов является одной из фундаментальных в области управления вычислительными системами и облачной инфраструктурой. Стремительный

рост масштабов облачных платформ, увеличение числа конкурирующих задач и гетерогенность доступных вычислительных ресурсов обусловливают необходимость разработки эффективных механизмов, способных принимать обоснованные решения в условиях неопределенности. При этом классические подходы к оптимизации, основанные на максимизации или минимизации единственной целевой функции, оказываются недостаточными, поскольку реальные системы распределения ресурсов характеризуются множеством конфликтующих показателей качества, таких как время выполнения задач, балансировка нагрузки, стоимость использования ресурсов, энергопотребление и т.п.

В данном контексте многокритериальная оптимизация и, в частности, концепция оптимальности по Парето, приобретает особую практическую значимость. Понятие Парето-оптимальности, введенное итальянским экономистом и социологом Вильфредо Парето в конце XIX века [1], позволяет формализовать компромисс между несовместимыми критериями без введения субъективных весовых коэффициентов. Решение называется Парето-оптимальным, если не существует другого допустимого решения, которое было бы не хуже по всем критериям и строго лучше хотя бы по одному. Совокупность таких решений образует Парето-множество (Парето-фронт), предоставляя лицу, принимающему решения, полный спектр компромиссных альтернатив.

Предсказательное распределение ресурсов представляет собой подход, при котором решения о назначении вычислительных ресурсов принимаются на основе прогнозирования показателей обработки задачи, построенного по историческим данным о загрузке системы [2]. Такой подход позволяет проактивно реагировать на изменения нагрузки, снижать задержки при выделении ресурсов и повышать общую эффективность использования инфраструктуры. Интеграция многокритериальной Парето-оптимизации в механизм предсказательного распределения ресурсов обеспечивает возможность одновременного учета нескольких показателей качества при формировании множества кандидатов на выделение ресурсов, что существенно повышает гибкость и обоснованность принимаемых решений.

Вместе с тем применение Парето-оптимизации в системах реального времени сопряжено с серьезной вычислительной проблемой. Базовый алгоритм построения Парето-множества основан на попарном сравнении всех кандидатов по каждому из рассматриваемых критериев. При увеличении числа кандидатов, характерном для крупных облачных платформ, данная зависимость становится узким местом системы, ограничивая применимость Парето-оптимизации в условиях жестких временных ограничений. Указанная проблема обусловливает необходимость исследования и применения методов снижения вычислительной сложности данного этапа.

Вопросы повышения эффективности алгоритмов Парето-оптимизации нашли отражение в ряде научных работ. Алгоритм Кунга [3], основанный на стратегии "разделяй и властвуй", позволил снизить сложность нахождения максимальных векторов до O(n log n) для двухкритериального случая. Широко применяемый алгоритм NSGA-II [4, 5] ввел эффективную процедуру недоминируемой сортировки с применением операции сравнения рангов, ставшую основой для последующих разработок. В работе [6] предложен алгоритм ENS (Efficient Non-dominated Sorting), достигающий существенного ускорения за счет сокращения числа выполняемых сравнений. Такие алгоритмы, как ENS и другие методы недоминируемой сортировки, подробно рассмотрены в обзорных работах [7], где они классифицированы по принципам построения. Алгоритмы Rank Sort [8, 9] используют битовый параллелизм и пересечения множеств, обеспечивая дополнительное ускорение по сравнению с предшествующими методами. С. Е. Кривобокова и В. А. Родин в своей статье [10] затрагивают алгоритмические сложности построения множества Парето, связанные с конфигурацией массива точек, и предлагают авторский алгоритм для

их решения. Тем не менее применение указанных подходов в контексте предсказательного распределения ресурсов в облачных системах остается недостаточно исследованным.

Целью настоящей статьи является исследование вычислительной сложности построения Парето-множества в задаче многокритериального предсказательного распределения ресурсов и анализ методов ее снижения. Для достижения поставленной цели решаются следующие задачи:

1) построить и исследовать зависимости алгоритмической сложности базового алгоритма построения Парето-множества от основных параметров: количества узлов-кандидатов и числа критериев;

2) проанализировать известные подходы к снижению алгоритмической сложности построения Парето-множества, оценить вычислительную сложность построения Парето-множества с их применением;

3) провести экспериментальное исследование на программной имитационной модели для сравнения ранее рассмотренных подходов между собой и с базовым алгоритмом;

4) предложить условия эффективного применения рассмотренных подходов к снижению алгоритмической сложности построения Парето-множества в задачах распределениях ресурсов вычислительных систем. Решение поставленных задач позволит обоснованно выбирать алгоритм построения Парето-множества в зависимости от масштаба системы и требований к времени отклика планировщика.

Метод предсказательного распределения ресурсов в вычислительных системах

Настоящая статья является продолжением работы [11], в которой был предложен предсказательный метод распределения ресурсов в вычислительных системах на основе многокритериальной модели принятия решений. В указанной работе один из ключевых этапов метода – построение множества Парето-оптимальных решений (P), однако вычислительная сложность данного этапа и возможности ее снижения не исследовались. Настоящая статья восполняет этот пробел. Для обеспечения преемственности на рис. 1 приводится формализации системы в объеме, необходимом для дальнейшего анализа.

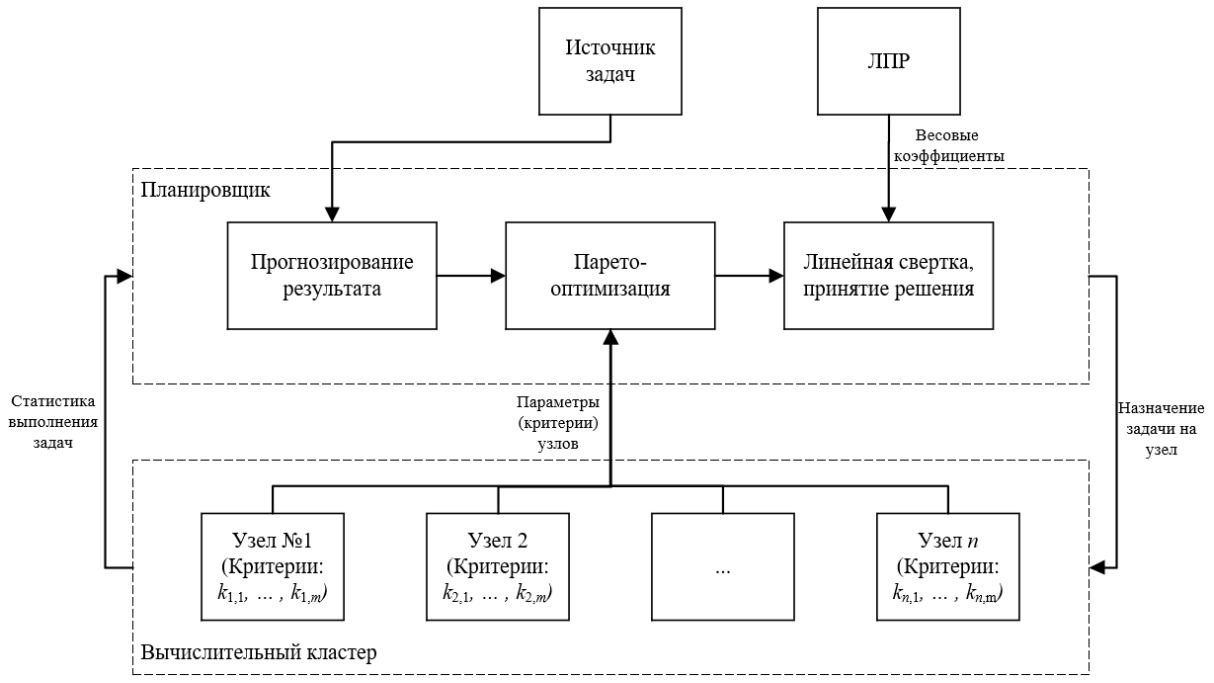

Рис. 1. *Система распределения ресурсов*

Построение множества P реализовано методом полного попарного сравнения. Вычислительная сложность такого алгоритма определяется следующим образом. Число пар узлов, подлежащих сравнению, составляет $\frac{n\cdot(n-1)}{2}$, где n – размер множества кандидатов. Для каждой пары выполняется сравнение по m критериям. Тогда, общее число элементарных операций сравнения равно:

$$\frac{n \cdot (n-1)}{2} \cdot m \ \rightarrow O(n^2 \cdot m).$$

При небольшом числе узлов данная сложность не является критичной. Однако в реальных вычислительных системах число кандидатов n может быть существенно больше: в крупных облачных кластерах количество доступных узлов достигает сотен и тысяч. В таких условиях квадратичная зависимость от n становится узким местом алгоритма, ограничивая применимость Парето-оптимизации в системах реального времени.

Кроме того, в задачах предсказательного распределения ресурсов построение Парето-множества выполняется при поступлении каждой новой задачи Z. При высокой интенсивности входящего потока задач суммарные затраты на этап Парето-оптимизации могут стать доминирующей составляющей общей вычислительной нагрузки планировщика.

Базовый алгоритм построения Парето-множества

Базовый алгоритм построения Парето-множества основан на методе полного попарного сравнения всех узлов-кандидатов. Он принимает на вход множество узлов $U = \{u_1, u_2, \ldots, u_n\}$ с вычисленными кортежами критериев $C(u_j, Z)$ и возвращает подмножество $P \subseteq U$, содержащее все недоминируемые узлы:

$$D_{\text{базовый}}(n, m) = \ O(n^2 \cdot \ m).$$

Алгоритм 1. Первоначально все узлы считаются кандидатами в Парето-множество. Для каждой упорядоченной пары (u_i, u_j) выполняется проверка: доминирует ли u_i узел u_j. Если по результатам сравнения всех m критериев u_i не хуже u_j по всем критериям и строго лучше хотя бы по одному, узел u_j исключается из P. По завершении двойного цикла в P остаются только недоминируемые узлы.

Зависимость от параметров. При фиксированном числе критериев m сложность алгоритма растет квадратично по числу узлов n. При фиксированном n сложность растет линейно по m. На практике m, как правило, невелико (в рассматриваемой задаче m = 3), поэтому определяющим параметром является n. В таблице 1 приведены оценки числа операций сравнения для разных значений n при m = 3.

Таблица 1. *Число операций сравнения базового алгоритма при m = 3*

n узлов	$O(n^2 \cdot m)$
10	270
50	7 350
100	29 700
500	747 000
1000	2 997 000
5000	74 970 000

Как видно из табл.1, уже при n = 500 число операций превышает 700 тысяч, а при n = 5 000 достигает порядка 75 миллионов. При высокой интенсивности входящего потока задач, когда Парето-множество строится при поступлении каждой новой задачи, суммарные затраты могут стать критичными.

Оценка времени выполнения. Пусть τ – среднее время выполнения одной элементарной операции сравнения на целевой вычислительной платформе. Тогда ожидаемое время построения Парето-множества базовым алгоритмом:

$$T_{\text{базовый}}(n, m) = \tau \cdot n(n-1) \cdot m.$$

При известной интенсивности входящего потока задач λ (задач в единицу времени) среднее процессорное время, затрачиваемое на этап Парето-оптимизации в единицу времени:

$$L = \lambda \cdot \tau \cdot n(n-1) \cdot m.$$

Условие применимости базового алгоритма в системе реального времени: $L < 1$, то есть суммарная нагрузка от этапа Парето-оптимизации не должна превышать пропускную способность одного процессорного ядра. При нарушении этого условия требуется применение методов снижения вычислительной сложности.

Методы ускорения построения Парето-множества

Снижение вычислительной сложности при построении Парето-фронта является актуальной задачей. Один из подходов, называемый "прореживанием" (pruning), предполагает сокращение множества решений еще до или в процессе сравнения. Обзор таких методов представлен в работе [12], где они классифицированы по способу отбора репрезентативных точек.

Предварительная фильтрация альтернатив по ключевому критерию

Идея метода состоит в том, чтобы до начала попарного сравнения сократить размер множества кандидатов за счет удаления узлов, заведомо не способных войти в Парето-множество. Для этого выбирается один ключевой критерий $C_{\text{ключ}}$ наиболее значимый с точки зрения задачи распределения – и по нему вводится пороговый фильтр.

В контексте предсказательного распределения ресурсов в качестве ключевого критерия естественно выбрать $C_{\text{ключ}}$ прогнозное время выполнения задачи, поскольку именно оно непосредственно определяет быстродействие системы. Фильтрация выполняется следующим образом: вычисляется минимальное значение ключевого критерия по всем узлам, после чего из рассмотрения исключаются узлы, значение $C_{\text{ключ}}$ которых превышает заданный порог $\theta \cdot \min(C_{\text{ключ}})$.

Здесь параметр θ регулирует жесткость фильтрации: при $\theta = 1$ сохраняются только узлы с минимальным значением $C_{\text{ключ}}$, при $\theta \to \infty$ фильтрация вырождается в базовый алгоритм. На практике θ выбирается в зависимости от допустимой потери качества решения.

Алгоритм 2. Предварительная фильтрация по ключевому критерию.

Вход: множество узлов U, кортежи критериев $C(u_j)$, ключевой критерий $C_{\text{ключ}}$, порог $\theta \geq 1$.

Выход: отфильтрованное множество $U^* \subseteq U$, Парето-множество P.

Здесь базовый алгоритм, применяемый к отфильтрованному множеству U^*.

Анализ вычислительной сложности. Пусть после фильтрации в U^* остается n' узлов, где $n' = \alpha \cdot n$, $\alpha \in (0, 1]$ – коэффициент сохранения. Тогда сложность метода:

$$D_{\text{фильтр}}(n, m) = O(n) + O(n'^2 \cdot m) = O(n) + O(\alpha^2 n^2 \cdot m).$$

Первое слагаемое – однократный проход по U для вычисления минимума и формирования U^*. Второе слагаемое – базовый алгоритм на U^*.

При $\alpha < 1$ достигается ускорение относительно базового алгоритма:

$$S_{\text{фильтр}} = \frac{D_{\text{базовый}}}{D_{\text{фильтр}}} \approx \frac{n^2}{n'^2} = \frac{1}{\alpha^2}.$$

Например, если фильтрация отсеивает половину узлов ($\alpha = 0{,}5$), ускорение составляет приблизительно в 4 раза. Если отсеивается 80% узлов ($\alpha = 0{,}2$), ускорение достигает 25 раз.

Корректность метода. Метод гарантирует включение в P всех истинно Парето-оптимальных узлов при условии, что ни один из них не был отфильтрован. Это условие выполняется, если θ выбрано достаточно большим. При слишком жестком пороге (малом θ) возможно исключение некоторых Парето-оптимальных узлов, поэтому выбор θ является компромиссом между скоростью и полнотой результата.

Ранняя остановка при попарном сравнении

В базовом алгоритме для каждой пары (u_i, u_j) сравнение всегда выполняется по всем m критериям, даже если факт отсутствия доминирования может быть установлен раньше. Метод ранней остановки модифицирует внутренний цикл по критериям: как только в процессе проверки обнаруживается, что доминирование невозможно, сравнение пары прекращается досрочно.

Условие досрочного завершения: если при проверке критерия C_k обнаружено, что $C_k(u_i) > C_k(u_j)$, то узел u_i не может доминировать u_j (поскольку нарушено условие – не хуже по всем критериям). Дальнейшая проверка оставшихся критериев для данной пары не имеет смысла.

Алгоритм 3. Построение Парето-множества с ранней остановкой.

Вход: множество узлов U, кортежи критериев $C(u_j)$, число критериев m.

Выход: Парето-множество P.

Отличие от базового состоит исключительно в определении условий доминирования. При обнаружении первого нарушения данных условий, цикл проверки по критериям прерывается.

Анализ вычислительной сложности. Худший случай остается прежним: $O(n^2 \cdot \mathrm{m})$. Он реализуется, когда все узлы несравнимы (никто не доминирует никого) и нарушение условия доминирования обнаруживается лишь на последнем критерии.

Средний случай. Пусть критерии проверяются в некотором порядке. Если значения критериев распределены независимо и равномерно, то вероятность того, что узел u_i не хуже u_j по первому критерию, равна приблизительно 0.5. Ожидаемое число проверяемых критериев до обнаружения нарушения:

$$E[k_{\text{стоп}}] = \sum_{k=1}^{m} k \cdot 0{,}5^{k-1} \cdot 0{,}5 \approx 2.$$

В среднем для каждой пары проверяется около 2 критериев вместо m. Ожидаемое число операций:

$$T_{\text{ранний выход}}(n, m) \approx n(n-1) \cdot E[k_{\text{стоп}}] = O(n^2),$$

при $m >> 2$, что дает преимущество в $m/2$ раз по сравнению с базовым алгоритмом.

Влияние порядка критериев. Эффективность ранней остановки существенно зависит от порядка проверки критериев. Наибольшее ускорение достигается, если первым проверяется критерий с наибольшей дискриминирующей силой – то есть тот, по которому узлы чаще всего различаются. В задаче предсказательного распределения ресурсов таким критерием может являться текущая загрузка узла (занимаемое процессорное время в данный момент) или прогнозное время выполнения задачи, поскольку производительность узлов существенно варьируется в гетерогенных системах.

Корректность метода. Ранняя остановка не изменяет результат алгоритма: множество P, возвращаемое алгоритмом 3, идентично множеству P алгоритма 1. Метод является точным и не вносит никакого приближения.

Инкрементальное построение Парето-множества

В предыдущих двух методах Парето-множество строится заново при каждом вызове алгоритма. Инкрементальный подход применим в ситуации, когда множество кандидатов U пополняется поэлементно – например, когда характеристики узлов обновляются последовательно по мере поступления данных о текущей загрузке.

Идея метода: поддерживать актуальное Парето-множество P, добавляя в него новые узлы по одному. При поступлении нового узла $u_{нов}$ выполняется его проверка относительно текущего P:

- если $u_{нов}$ доминируется хотя бы одним элементом P, он не включается в P;
- если $u_{нов}$ доминирует некоторые элементы P, они исключаются из P, а $u_{нов}$ добавляется;
- если $u_{нов}$ несравним со всеми элементами P, он добавляется в P без исключений.

Алгоритм 4. Инкрементальное построение Парето-множества.

Вход: новый узел $u_{нов}$, текущее Парето-множество P, число критериев m.

Выход: обновленное Парето-множество P.

Полное построение Парето-множества с нуля выполняется последовательным применением алгоритма 4 ко всем n узлам:

Анализ вычислительной сложности:

Однократное добавление узла. При добавлении $u_{нов}$ в P размером $|P| = p$ выполняется не более p сравнений (по одному на каждый элемент P), каждое из которых включает не более m проверок критериев. Сложность одного шага:

$$T_{инк.шаг} = O(p \cdot m).$$

Полное построение с нуля. При последовательном добавлении n узлов размер P в общем случае изменяется. В лучшем случае $|P| = 1$ на каждом шаге (один узел доминирует всех предыдущих) – сложность: $O(n \cdot m)$.

В худшем случае $|P|$ растет до n (все узлы несравнимы) – сложность:

$$T_{инк} = \sum_{i=1}^{n} i \cdot m = m \cdot \frac{n(n+1)}{2} = O(n^2 \cdot m),$$

что совпадает с базовым алгоритмом. При этом асимптотика в худшем случае не улучшается, однако на практике $|P| << n$, и метод значительно эффективнее.

Главное преимущество инкрементального подхода проявляется при динамическом обновлении множества узлов. Если в уже построенное Парето-множество P добавляется один новый узел (например, после обновления загрузки узла), стоимость обновления составляет $O(|P| \cdot m)$, а не $O(n^2 \cdot m)$. В задаче предсказательного распределения ресурсов это особенно актуально: при поступлении новой задачи Z изменяется загрузка одного узла (того, на который была назначена предыдущая задача), и Парето-множество требует лишь локального обновления.

Инкрементальные принципы построения Парето-множества также исследуются в контексте иерархических методов сортировки, например, в работе [13], где авторы анализируют и улучшают алгоритм HNDS, обеспечивающий предсказуемую сложность в худшем случае.

Экспериментальное исследование

Для экспериментального подтверждения теоретических оценок вычислительной сложности и сравнения рассматриваемых алгоритмов была разработана программа моделирования на языке Python. Эксперименты проводились на вычислительной платформе со следующими характеристиками: процессор Ryzen 5 5600, ОЗУ 32 Gb, Windows 11.

Генерация тестовых данных осуществлялась следующим образом. Для каждого эксперимента формировалось множество узлов U размером n, для каждого узла u_j случайным образом генерировался кортеж критериев $C(u_j) = \langle c_1, c_2, c_3 \rangle$. Значения каждого критерия выбирались из логнормального распределения на отрезке $[c_{min}, c_{max}]$.

Измерялось время выполнения каждого алгоритма в миллисекундах. Для устранения случайных флуктуаций каждый эксперимент повторялся $R = 30$ раз, после чего вычислялось среднее значение времени выполнения. Корректность всех алгоритмов проверялась сравнением возвращаемого ими множества P с эталонным результатом базового алгоритма.

Сравнение алгоритмов в зависимости от числа узлов *n*

В первой серии экспериментов исследовалось поведение алгоритмов от числа узлов-кандидатов n при фиксированном $k = 3$ и параметре фильтрации $\theta = 3.0$. Значение n варьировалось в диапазоне от 10 до 1000.

Результаты первой серии экспериментов представлены на рис. 2 и 3, где показана зависимость среднего времени выполнения и среднего количества операций алгоритмов от n.

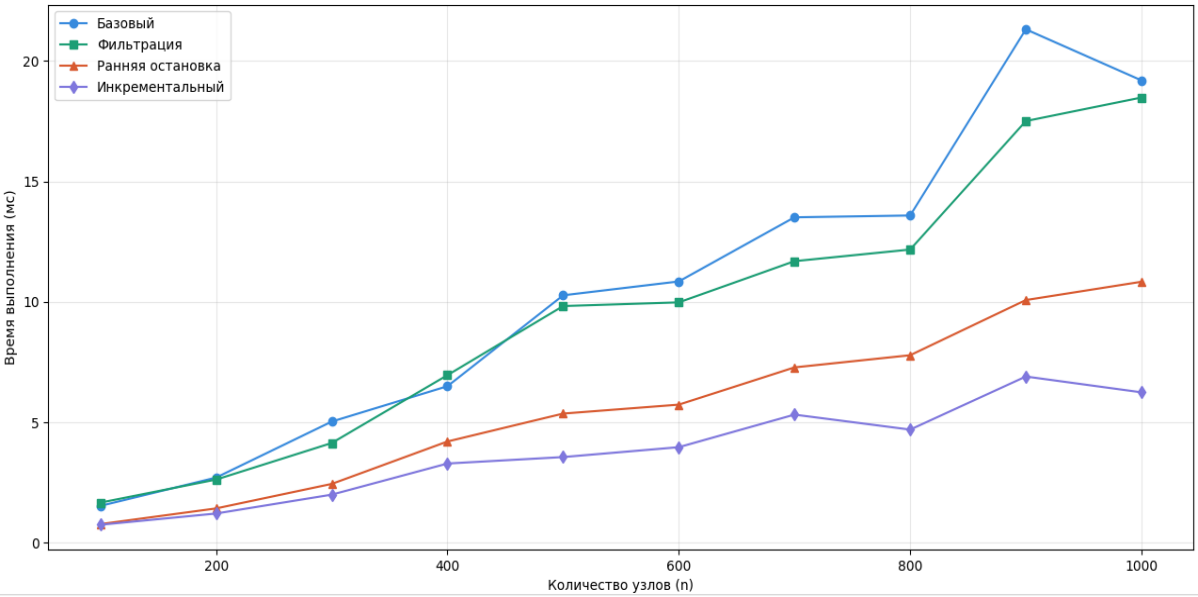

Рис. 2. *Среднее время выполнения алгоритмов (мс) при k = 3*

Как видно из рисунка 2, все четыре алгоритма демонстрируют квадратичный рост времени выполнения при увеличении n, что соответствует теоретическим оценкам $O(n^2 \cdot m)$. Однако абсолютные значения времени и темп роста существенно различаются.

а)

б)

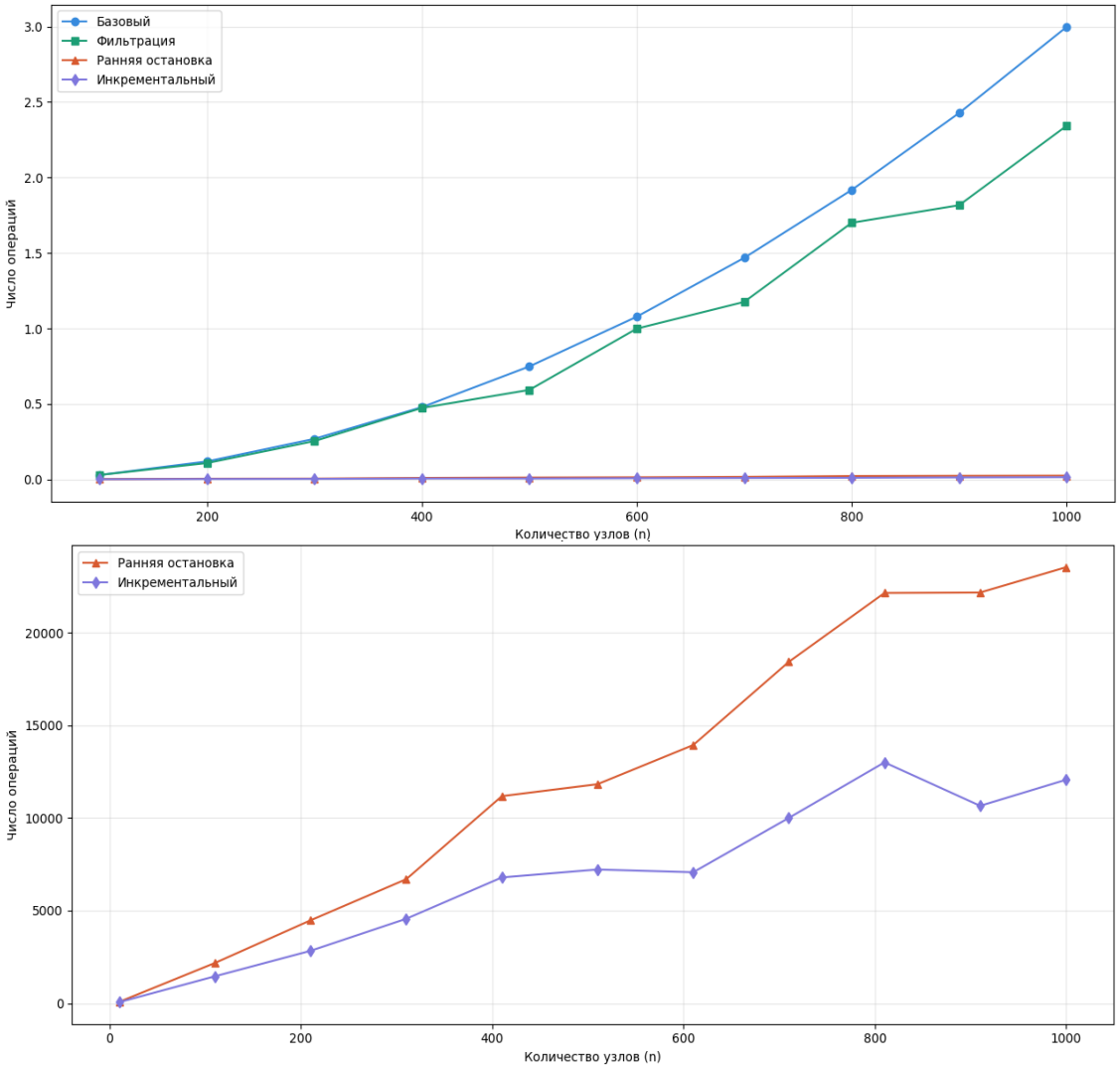

Рис. 3. *Среднее количество операций при k = 3, а) график со всеми алгоритмами, б) график без предварительной фильтрации и базового алгоритмов*

Результаты эксперимента демонстрируют следующее. Базовый алгоритм показывает выраженный квадратичный рост времени выполнения: от 0.05 мс при $n = 10$ до 21.79 мс при $n = 1000$, что соответствует теоретической оценке $O(n^2 \cdot k)$.

Метод ранней остановки обеспечивает стабильное ускорение относительно базового на всем исследованном диапазоне – в среднем в 1.84× с незначительными отклонениями от точки к точке. При $n = 1000$ время выполнения составляет 13.58 мс. Стабильность ускорения обусловлена тем, что при логнормальном распределении значений критериев ожидаемое число проверяемых критериев до досрочного завершения остается постоянным и не зависит от n.

Инкрементальный алгоритм демонстрирует наибольшее ускорение среди рассматриваемых подходов и, в отличие от метода ранней остановки, это преимущество возрастает с увеличением n: от ×1.4–1.8 при малых n до ×2.9–3.9 при $n \geq 900$. При $n = 1000$ время выполнения составляет 7.58 мс – более чем в 2.9 раза быстрее базового. Рост ускорения объясняется тем, что с увеличением n размер Парето-множества $|P|$ растет медленнее, чем n, что снижает среднее число сравнений на каждом шаге инкрементального добавления.

Метод предварительной фильтрации в данном эксперименте не обеспечивает за-

метного ускорения: среднее ускорение составляет лишь ×1.07, а в ряде точек метод незначительно уступает базовому алгоритму по времени. Причина состоит в характере генерируемых данных: первый критерий c_1 имеет логнормальное распределение с выраженной правосторонней асимметрией, поэтому при $\theta = 3.0$ порог фильтрации захватывает значительную часть узлов, и размер отфильтрованного множества U^* лишь незначительно меньше исходного, 90–97% от U (рис. 4).

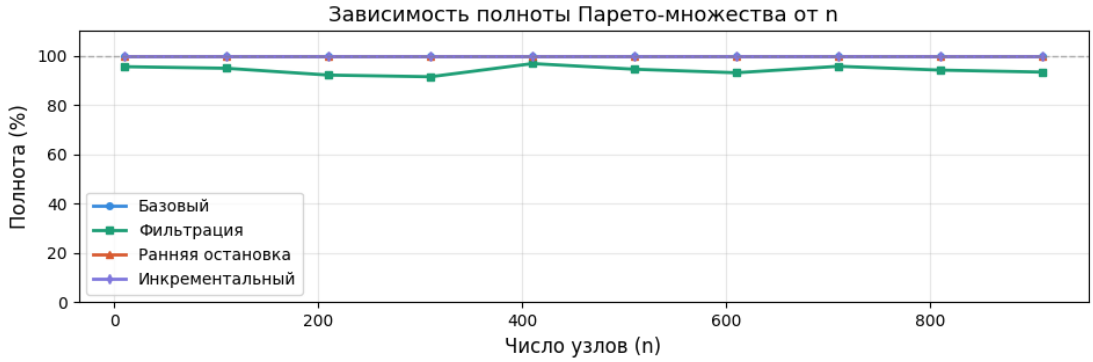

Рис. 4. *Зависимость полноты Парето-множества от количества узлов*

Таким образом, в условиях логнормального распределения критериев и при $\theta = 3.0$ наиболее эффективным методом является инкрементальный алгоритм, обеспечивающий ускорение до ×3.9 без потери полноты P. Метод ранней остановки гарантирует стабильное двукратное ускорение при 100% полноте и применим без каких-либо ограничений на характер данных. Метод предварительной фильтрации требует тщательной настройки параметра θ с учетом распределения значений ключевого критерия – при неудачном выборе θ он не дает преимущества ни по скорости, ни по полноте.

Сравнение алгоритмов в зависимости от числа критериев *k*

Во второй серии экспериментов исследовалась зависимость времени выполнения от числа критериев k при фиксированном $n = 200$. Значение k варьировалось от 2 до 10. Результаты экспериментов продемонстрированы на рис. 5.

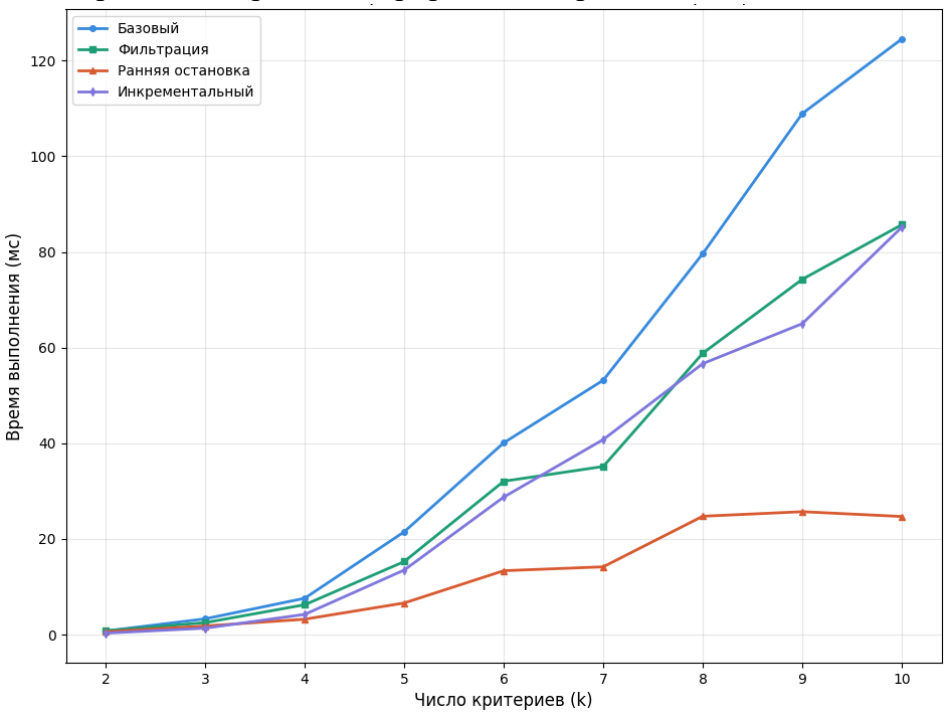

Рис. 5. *Среднее время выполнения алгоритмов (мс) при n = 200*

При увеличении числа критериев k наиболее эффективным методом становится ранняя остановка, преимущество которой монотонно возрастает. Инкрементальный метод целесообразно применять при малых k ($k \leq 4$), причина состоит в том, что с увеличением k размер Парето-множества $|P|$ растет – при большем числе критериев доминирование устанавливается реже, и в Парето-фронт попадает большая доля узлов. Это увеличивает число сравнений при каждом инкрементальном шаге, сильно снижая преимущество метода.

Влияние параметра θ на эффективность фильтрации

В третьей серии экспериментов исследовалось влияние порогового параметра θ на эффективность метода предварительной фильтрации. Фиксировались $n = 500$, $k = 3$; параметр θ варьировался от 1.0 до 5.0 с шагом 0.1.

Результаты эксперимента наглядно демонстрируют компромисс между вычислительной эффективностью и полнотой Парето-множества, характерный для метода предварительной фильтрации (рис. 6).

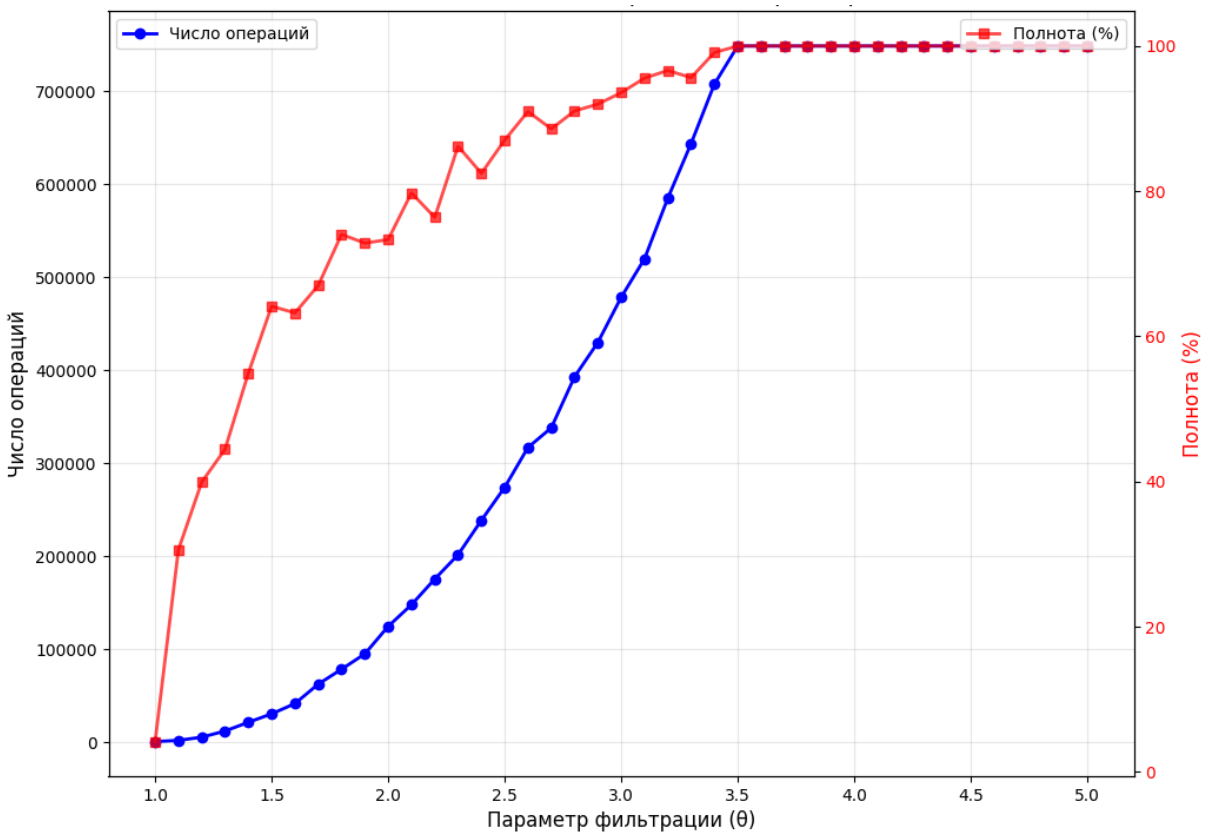

Рис. 6. *Влияние параметра θ на эффективность алгоритма фильтрации*

При $\theta = 1.0$ метод обеспечивает максимальное ускорение по числу операций (×1497 относительно базового), однако полнота P составляет лишь 1.8% – подавляющее большинство Парето-оптимальных узлов отсекается. С увеличением θ размер отфильтрованного подмножества U^* и число операций монотонно возрастают, а полнота в целом улучшается, хотя и с заметными флуктуациями, обусловленными случайным характером генерируемых данных.

Можно выделить три характерных диапазона θ. При $\theta \in [1.0, 2.0]$ полнота остается низкой (до 76%), ускорение высокое (от ×6 до ×1497), однако результат нельзя считать практически применимым из-за значительных потерь Парето-оптимальных решений.

При $\theta \in [2.0, 3.3]$ полнота достигает приемлемого уровня: значение 90% впервые достигается при $\theta = 2.7$, а 97% – при $\theta = 3.1$, при этом ускорение по операциям составляет ×2.2 и ×1.2 соответственно. Начиная с $\theta = 3.4$ полнота стабилизируется на уровне 100% – фильтр перестает отсекать какие-либо Парето-оптимальные узлы. При $\theta \geq 3.5$ в отфильтрованное множество попадают все 500 узлов, и метод вырождается в базовый алгоритм, не давая никакого преимущества.

Таким образом, выбор параметра θ определяет баланс между точностью и эффективностью. Эта дилемма – "точность против эффективности" – является общей для многих методов аппроксимации в многокритериальной оптимизации. Детальный обзор таких подходов представлен в работе [14].

Заключение

В настоящей работе проведено исследование алгоритмической сложности построения Парето-множества применительно к задаче распределения ресурсов в вычислительных системах.

По результатам работы сформулированы следующие выводы:

1. Установлено, что базовый алгоритм полного попарного сравнения имеет вычислительную сложность $O(n^2 \cdot k)$, что при большом числе узлов-кандидатов и высокой интенсивности входящего потока задач может стать доминирующей составляющей нагрузки планировщика. Введено условие применимости базового алгоритма в системе реального времени через параметр загрузки $L < 1$.

2. Рассмотрены три подхода снижения алгоритмической сложности. Метод предварительной фильтрации сокращает размер множества кандидатов до начала попарного сравнения, достигая ускорения порядка $1/\alpha^2$ при коэффициенте сохранения α, однако сопряжен с возможными потерями полноты Парето-множества. Метод ранней остановки не изменяет результат алгоритма, снижая среднее число проверяемых критериев до ~2 при любом k. Инкрементальный метод обеспечивает ускорение при динамическом обновлении множества узлов, сводя стоимость локального обновления к $O(|P| \cdot k)$.

3. Экспериментально подтверждено, что при увеличении числа узлов n наибольшее и возрастающее ускорение обеспечивает инкрементальный алгоритм при гарантированной полноте P. При увеличении числа критериев k наиболее эффективной оказывается ранняя остановка, ускорение которой монотонно возрастает. Инкрементальный метод предпочтителен при малых k, когда размер Парето-фронта невелик.

4. Для метода предварительной фильтрации экспериментально определен практически целесообразный диапазон порогового параметра для заданной конфигурации имитационной модели, при котором полнота Парето-множества составляет не менее 90% при сохранении умеренного ускорения. При $\theta \geq 3.5$ метод вырождается в базовый алгоритм, не давая никакого преимущества.

Практическая значимость полученных результатов состоит в том, что они позволяют обоснованно выбирать алгоритм построения Парето-множества в зависимости от масштаба системы, числа критериев и допустимого уровня полноты полученного множества. В частности, для задач предсказательного распределения ресурсов с динамически обновляемой загрузкой узлов рекомендуется инкрементальный алгоритм; при необходимости гарантированной точности и произвольном характере данных – метод ранней остановки; при возможности предварительной оценки ключевого критерия и допустимости частичных потерь полноты – метод фильтрации с предварительно откалиброванным значением θ.

Направлениями дальнейших исследований являются: анализ вычислительной сложности при использовании комбинаций рассмотренных методов; исследование поведения алгоритмов при числе критериев $k > 10$ и большом числе Парето-оптимальных решений; экспериментальная проверка на реальных данных о загрузке вычислительных

кластеров. Также перспективным направлением является комбинирование разработанных методов с современными алгоритмами недоминируемой сортировки, например, с SETNDS [15], что может дополнительно повлиять на производительность.

Список источников

1. *Подиновский В. В., Ногин В. Д.* Парето-оптимальные решения многокритериальных задач: 2-е изд., испр. и доп. М.: ФИЗМАТЛИТ, 2007. 256 с. ISBN 978-5-9221-0812-6.
2. *Трушкин Е. С., Фрейман В. И.* Предсказательный метод распределения ресурсов в вычислительных системах // Информационные технологии и вычислительные системы. № 1. С. 133−144. 2026. DOI 10.14357/20718632260112. EDN TTXDFB.
3. *Kung H. T., Luccio F., Preparata F. P.* On Finding the Maxima of a Set of Vectors // Journal of the Association for Computing Machinery. 1975. Vol. 22, № 4. P. 469–476. URL: https://dl.acm.org/doi/10.1145/321906.321910 (дата обращения: 13.04.2026).
4. *Deb K., Pratap A., Agarwal S., Meyarivan T.* A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation. 2002. Vol. 6, № 2. P. 182−197. URL: https://doi.org/10.1109/4235.996017 (дата обращения: 13.06.2026).
5. *Biscani F., Izzo D.* A parallel global multiobjective framework for optimization: pagmo // Journal of Open Source Software. 2020. Vol. 5, Iss. 53. P. 2338. URL: https://doi.org/10.21105/joss.02338 (дата обращения: 13.04.2026).
6. *Zhang X., Tian Y., Cheng R., Jin Y.* An Efficient Approach to Nondominated Sorting for Evolutionary Multiobjective Optimization // IEEE Transactions on Evolutionary Computation. 2015. Vol. 19, № 2. P. 201–213. URL: https://doi.org/10.1109/TEVC.2014.2308305 (дата обращения: 13.04.2026).
7. *Kong. L., Pan. JS., Snášel. V.* A Review of Non-dominating Sorting Algorithms // Smart Innovation, Systems and Technologies. 2024. Vol. 394. P. 173−183. URL: https://doi.org/10.1007/978-981-97-3980-6_15 (дата обращения 14.04.2026).
8. *Zhou Y., Chen Z., Zhang J.* Ranking Vectors by Means of the Dominance Degree Matrix // IEEE Transactions on Evolutionary Computation. 2017. Vol. 21, № 1. P. 34–51. URL: https://doi.org/10.1109/TEVC.2016.2567648 (дата обращения: 13.04.2026).
9. *Martyniuk T. B., Krukivskyi B. I.* Advanced Model of Parallel Sorting Algorithm with Ranking // Cybernetics and Systems Analysis. 2024. Vol. 60. P. 45–49. URL: https://doi.org/10.1007/s10559-024-00645-y (дата обращения: 19.05.2026).
10. *Кривобокова С. Е., Родин В. А.* Алгоритм и программа для графического выделения множества Парето в точечном массиве // Прикладная математика & Физика. 2021. Т. 53, № 2. С. 125−131. DOI 10.52575/2687-0959-2021-53-2-125–131. URL: https://maths-physics-journal.ru/index.php/journal/article/view/59 (дата обращения 14.04.2026).
11. *Трушкин Е. С., Фрейман В. И.* Предсказательный метод распределения ресурсов в вычислительных системах на основе многокритериальной модели принятия решений // Информатика и автоматизация. Т. 25, № 2. С. 568−601. 2026. URL: https://doi.org/10.15622/ia.25.2.10 (дата обращения 12.04.2026).
12. *Petchrompo S., Coit D. W., Brintrup A., et al*. A review of Pareto pruning methods for multi-objective optimization // Computers & Industrial Engineering. 2022. Vol. 167, Iss. 1. P. 108022. URL: https://doi.org/10.1016/j.cie.2022.108022 (дата обращения 14.04.26).
13. *Prakash V., Mishra S.* Hierarchical non-dominated sort: analysis and improvement // Genetic Programming and Evolvable Machines. 2024. Vol. 25, № 1. Art. no 14. URL: https://doi.org/10.1007/s10710-024-09487-1 (дата обращения 14.04.2026).

14. *Herzel A., Ruzika S., Thielen C.* Approximation Methods for Multiobjective Optimization Problems: A Survey // INFORMS Journal on Computing. 2021. Vol. 33, № 4. P. 1280–1299. URL: https://doi.org/10.1287/ijoc.2020.1028 (дата обращения 14.04.2026).
15. *Zhou Y., et al*. SETNDS: A SET-Based Non-Dominated Sorting Algorithm for Multi-Objective Optimization Problems // Applied Sciences. 2020. Vol. 10, № 19. Art. No 6820. URL: https://doi.org/10.3390/app10196858 (дата обращения 14.04.2026).

References

1. Podinovsky, V. V. and Nogin, V. D. (2007), *Pareto-optimal'nye resheniya mnogokriterial'nykh zadach* [Pareto-optimal solutions of multicriteria problems], 2nd ed., revised and enlarged, FIZMATLIT, Moscow, 256 p. ISBN 978-5-9221-0812-6.
2. Trushkin, E. S. and Freiman, V. I. (2026), "Predskazatel'nyy metod raspredeleniya resursov v vychislitel'nykh sistemakh" [Predictive method of resource allocation in computing systems], *Informatsionnye tekhnologii i vychislitel'nye sistemy* [Information Technologies and Computing Systems], no 1, pp. 133–144, DOI 10.14357/20718632260112, EDN TTXDFB.
3. Kung, H. T., Luccio, F. and Preparata, F. P. (1975), "On finding the maxima of a set of vectors", *Journal of the Association for Computing Machinery*, vol. 22, no 4, pp. 469–476, URL: https://dl.acm.org/doi/10.1145/321906.321910 (accessed: 13.04.2026).
4. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002), "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, no 2, pp. 182–197, URL: https://doi.org/10.1109/4235.996017 (accessed: 13.06.2026).
5. Biscani, F. and Izzo, D. (2020), "A parallel global multiobjective framework for optimization: pagmo", *Journal of Open Source Software*, vol. 5, iss. 53, pp. 2338, URL: https://doi.org/10.21105/joss.02338 (accessed: 13.04.2026).
6. Zhang, X., Tian, Y., Cheng, R. and Jin, Y. (2015), "An efficient approach to nondominated sorting for evolutionary multiobjective optimization", *IEEE Transactions on Evolutionary Computation*, vol. 19, no 2, pp. 201–213, URL: https://doi.org/10.1109/TEVC.2014.2308305 (accessed: 13.04.2026).
7. Kong, L., Pan, J. S. and Snášel, V. (2024), "A review of non-dominating sorting algorithms", *Smart Innovation, Systems and Technologies*, vol. 394, pp. 173–183, URL: https://doi.org/10.1007/978-981-97-3980-6_15 (accessed: 14.04.2026).
8. Zhou, Y., Chen, Z. and Zhang, J. (2017), "Ranking vectors by means of the dominance degree matrix", *IEEE Transactions on Evolutionary Computation*, vol. 21, no 1, pp. 34–51, URL: https://doi.org/10.1109/TEVC.2016.2567648 (accessed: 13.04.2026).
9. Martyniuk, T. B. and Krukivskyi, B. I. (2024), "Advanced model of parallel sorting algorithm with ranking", *Cybernetics and Systems Analysis*, vol. 60, pp. 45–49, URL: https://doi.org/10.1007/s10559-024-00645-y (accessed: 19.05.2026).
10. Krivobokova, S. E. and Rodin, V. A. (2021), "Algoritm i programma dlya graficheskogo vydeleniya mnozhestva Pareto v tochechnom massive" [Algorithm and program for graphical highlighting of the Pareto set in a point array], *Prikladnaya matematika & Fizika* [Applied Mathematics & Physics], vol. 53, no 2, pp. 125–131, URL: https://maths-physics-journal.ru/index.php/journal/article/view/59 (accessed: 14.04.2026).
11. Trushkin, E. S. and Freiman, V. I. (2026), "Predskazatel'nyy metod raspredeleniya resursov v vychislitel'nykh sistemakh na osnove mnogokriterial'noy modeli prinyatiya resheniy" [Predictive method of resource allocation in computing systems based on a multi-criteria decision-making model], *Informatika i avtomatizatsiya* [Informatics and

Automation], vol. 25, no 2, pp. 568–601, URL: https://doi.org/10.15622/ia.25.2.10 (accessed: 12.04.2026).
12. Petchrompo, S., Coit, D. W., Brintrup, A. et al. (2022), "A review of Pareto pruning methods for multi-objective optimization", *Computers & Industrial Engineering*, vol. 167, iss. C, URL: https://doi.org/10.1016/j.cie.2022.108022 (accessed: 14.04.2026).
13. Prakash, V. and Mishra, S. (2024), "Hierarchical non-dominated sort: analysis and improvement", *Genetic Programming and Evolvable Machines*, vol. 25, no 1, art. no 14, URL: https://doi.org/10.1007/s10710-024-09487-1 (accessed: 14.04.2026).
14. Herzel, A., Ruzika, S. and Thielen, C. (2021), "Approximation methods for multiobjective optimization problems: a survey", *INFORMS Journal on Computing*, vol. 33, no 4, pp. 1280–1299, URL: https://doi.org/10.1287/ijoc.2020.1028 (accessed: 14.04.2026).
15. Zhou, Y. et al. (2020), "SETNDS: a SET-based non-dominated sorting algorithm for multi-objective optimization problems", *Applied Sciences*, vol. 10, no 19, art. no 6820, URL: https://doi.org/10.3390/app10196858 (accessed: 14.04.2026).

Сведения об авторах
Трушкин Е. С. − аспирант кафедры "Автоматика и телемеханика" Пермского национального исследовательского политехнического университета (614990, Россия, г. Пермь, Комсомольский пр., д. 29).

About the authors
Trushkin E. S. − a postgraduate student at the Department of Automation and Telemechanics at the Perm National Research Polytechnic University (29 Komsomolsky Prospekt, Perm, Russia, 614990).