Динамическое перепланирование расписания проектов по разработке программного обеспечения

М.И. Ардатовский, Т.Г. Максимова

Национальный исследовательский университет ИТМО, Санкт-Петербург

Аннотация: В статье проведён анализ современных методов планирования и перепланирования расписаний проектов по разработке программного обеспечения в условиях динамично изменяющихся требований и ограниченных ресурсов. Сформулирована научная задача оптимизации процессов управления командой разработки посредством динамического распределения работ, что является актуальным ввиду необходимости быстрого реагирования на изменения в производственном календаре и входных данных проекта. Предложенное решение основывается на использовании генетического алгоритма для автоматизации перепланирования расписания, позволяющего учитывать такие ключевые критерии эффективности как минимизацию простоев, оптимальное распределение нагрузки между сотрудниками и соблюдение технологических зависимостей между требованиями. Проведённый анализ показал, что методика динамичного перепланирования способна оперативно корректировать исходное расписание при появлении новых требований или изменении условий работы, что обеспечивает значительное улучшение качества планирования и повышает устойчивость системы к внешним изменениям. Результаты исследования подтвердили, что разработанный подход позволяет достичь оптимального баланса между полезностью выполненных работ и затратами на корректировку плана. Полученные данные свидетельствуют о высокой перспективности использования генетических алгоритмов для динамичного управления расписаниями в проектах по разработке программного обеспечения, что открывает возможности для дальнейших исследований в области гибких методов управления и адаптивного распределения ресурсов в условиях неопределённости.
Ключевые слова: генетические алгоритмы, задачи оптимизации, теория расписаний, управление проектами, проблема упорядочивания проектов с ограниченными ресурсами.

Введение

Задачи составления расписания занимают центральное место в управлении проектами, производстве, логистике и многих других сферах. Под составлением расписания понимают процесс распределения ограниченных ресурсов по набору задач или операций, чтобы минимизировать общее время выполнения, сократить издержки или достичь других ключевых целей.

Существуют различные постановки задач из теории расписаний, начиная от классических «Рабочий цех» и «Потоковая линия», заканчивая более сложными вариантами, учитывающими распределённые вычислительные среды и работу в реальном времени [1]. Тем не менее даже

такое разнообразие методов зачастую оказывается недостаточным в условиях современных, быстро меняющихся проектов, когда требования могут меняться достаточно быстро, а доступность ресурсов может колебаться.

Один из классических вариантов составления расписания использует набор работ, каждая из которых состоит из последовательности операций, требующих разных ресурсов. Целью составления расписания является минимизация времени завершения всех работ, расходов, задержек и пр. В работе [2] описывается историческая эволюция постановки и указывается, что «Рабочий цех» является NP-трудной задачей, в решении которой широко применяются эвристики, метаэвристики и методы математического программирования. В отличие от «Рабочего цеха», в задаче «Потоковая линия» ресурсы расположены в фиксированной последовательности, через которую проходят все работы. Каждая работа выполняется последовательно на каждом ресурсе, тогда как целью, как правило, является минимизация общего времени производства или запаздывания. В обзорной статье [3] рассмотрено множество вариаций «Задачи планирования для поточной линии», включая гибридные постановки, где может быть несколько параллельных машин на одном из этапов. Такие задачи встречаются, например, на сборочных конвейерах и в цехах серийного производства.

Более комплексная задача «Открытая линия» схожа с предыдущими, однако в ней отсутствует жёстко заданная очередность выполнения операций, то есть каждая операция может быть запланирована на любую машину и в любом порядке, лишь бы все операции конкретной задачи были выполнены [4]. В работе [5] подчёркивается сложность оптимального решения задачи «Открытая линия» из-за очень гибкой структуры работы часто приходится применять эвристические или метаэвристические методы.

При подходе, основанном на методах декларативного программирования и ограничения, задача формулируется в виде набора

переменных, доменов и ограничений, а поиск решения ведётся специализированными алгоритмами. Как отмечают авторы [6], методы «Программирования в ограничениях» хорошо подходят для решения сложных задач планирования и составления расписания с многочисленными зависимостями, например, временными окнами, прецедентными отношениями, уникальными ресурсами. «Программирования в ограничениях» может комбинироваться с эвристиками, чтобы ускорить поиск оптимального или близкого к оптимальному решения.

Среди подобных постановок наибольшую актуальность имеет «Проблема упорядочивания проектов с ограниченными ресурсами» (Resource-Constrained Project Scheduling Problem – RCPSP), которая является сложной задачей планирования и управления проектами, возникающая во многих областях, таких как строительство, инженерия, производство и ИТ-проекты [7]. Основная цель RCPSP это - эффективное распределение ресурсов для выполнения набора задач в рамках проекта в условиях ограниченных ресурсов.

Ключевым понятием RCPSP являются задачи, так, проект состоит из различных задач, каждая из которых имеет определенную продолжительность и зависимости от других задач. Задачи могут требовать одновременного выполнения или выполнения в определенной последовательности. Ресурсы могут быть ограничены и могут включать в себя трудовые ресурсы, оборудование, материалы или финансы. Каждая задача требует определенного количества ресурсов для выполнения. Ограничения могут быть связаны с доступностью ресурсов, временными рамками проекта и последовательностью задач. Например, определенная задача не может начаться, пока не будет завершена другая задача, или некоторые ресурсы могут быть доступны только в определенные периоды. Основная цель состоит

в том, чтобы минимизировать общее время выполнения проекта или другие показатели, такие как стоимость, при соблюдении всех ограничений.

Основной нерешенной задачей, относящийся к проектам по разработке программного обеспечения является подзадача RCPSP, которая подразумевает динамическое перепланирование составленного расписания в связи с изменениями во входных данных.

Цель исследования заключается в решении научной задачи оптимизации процесса перепланирования проектов по разработке программного обеспечения за счет реализации генетического алгоритма динамического перепланирования расписания сотрудников.

Задачами исследования являются:

- анализ методов планирования и перепланирования проектов, относящихся к задаче RCPSP;

- разработка генетического алгоритма динамического перепланирования расписания сотрудников, учитывающего изменения в производственном календаре и оценки задач;

- проведение экспериментальной проверки разработанного алгоритма с использованием общепринятых в данной предметной области метрик.

Литературный обзор

В первую очередь стоит рассмотреть обзорные исследования [8, 9]. В первом даётся всеобъемлющий обзор классических и современных подходов к RCPSP за последние годы. Авторы второго исследования концентрируются на многопроектном планировании, подробно классифицируя методы и рассматривая возникающие в таких задачах эффекты межпроектного взаимодействия. Данные систематические обзоры показывают, как эволюционируют методы решения RCPSP от ранних эвристик к гибридным метаэвристическим алгоритмам и моделям, учитывающим разные источники неопределённости.

В работе [10] отмечается новый тренд решения проблемы. Подчеркивается смещение фокуса исследований от чисто метаэвристических методов к гибридным метаэвристическим подходам при решении NP-трудной задачи RCPSP, имеющей широкое применение в таких областях, как производство, планирование производства и управление проектами. Также сравнивается эффективность этих методов на данных открытой библиотеки PSPLIB и подчеркивается их преимущества в решении сложных задач планирования.

Исследование [11] внесло значимый вклад в область RCPSP, так как в нем была предложена параллельная схема планирования и введены новые концепции, такие как вес ресурсов, связанных с задачей, и коэффициент сотрудничества. Данный эвристический метод показал высокую эффективность в быстром нахождении улучшенных решений.

Проблема RCPSP становится ещё более сложной, когда одновременно рассматривается несколько параллельно выполняемых проектов. В таких случаях речь идёт о мультипроектных постановках, требующих координации ресурсов и временных зависимостей между разными проектами. В исследовании [12] рассматриваются методы решения мультипроектного RCPSP с применением многомодового генетического алгоритма, что позволяет более гибко учитывать различные варианты выполнения задач. Авторы статьи [13] уделяют особое внимание созданию эталонных наборов данных и разработке «разделённого» подхода к расписанию, позволяющего эффективно управлять крупными портфелями проектов. В работе [14] представлена вариация задачи с учётом нескольких вариантов выполнения и более сложных ресурсных ограничений, где предложенные алгоритмы демонстрируют высокую эффективность на промышленных примерах. Наконец, в статье [15] вводится реактивная версия мультипроектного планирования при внезапном появлении нового проекта, подчёркивающая

важность механизмов быстрого перераспределения ресурсов и приоритетов. Аналогичная идея заложена и в исследовании [16], где анализируется реактивное планирование для многопроектной среды при динамическом появлении нового проекта.

Постановка задачи исследования

Для формальной постановки однопроектной версии задачи обозначим:

I — множество требований i пронумерованных $1, \dots, |\mathrm{I}|$. При этом между некоторыми парами требований могут быть заданы ограничения предшествования: $i \rightarrow j$ означает, что обслуживание требования j начинается не раньше окончания обслуживания требования i.

E — множество сотрудников e пронумерованных $1, \dots, |\mathrm{E}|$.

T — множество доступных дней t пронумерованных $1, \dots, |\mathrm{T}|$.

H — максимальное количество ресурсов (тип ресурса может меняться в зависимости от постановки)

h_{ie} — количество ресурсов, необходимых сотруднику e для обслуживания требования i.

s_i — время начала обслуживания требования i.

d_i — продолжительность выполнения обслуживания требования i.

$x_{ie} \in \{0,1\}$ — бинарная переменная, отображающая признак потребления ресурсов при обслуживании требования i сотрудником e.

При этом для расчета времени завершения проекта определяется время завершения самой поздней работы, тогда постановка отображается как:

$$\max_{i \in I}(s_i + d_i) \rightarrow min$$

при ограничениях на общий срок проекта (1), на общее количество ресурсов (2), неотрицательное время начала обслуживания требования (3), ограничение предшествования (4), обязательность назначения каждого обслуживания

требования (5), ограничение на то, что сотрудник в определенный день может обслуживать не более чем одно требование (6):

1. $s_i + d_i \leq T$
2. $\sum_{i \in I} \sum_{e \in E} h_{ie} x_{ie} \leq H$
3. $s_i \geq 0$
4. $s_j \geq s_i + d_i$
5. $\sum_{e \in E} x_{ie} = 1 \quad \forall\, i \in I$
6. $\sum_{i \in I} x_{iet} \leq 1 \quad \forall\, t \in T, \forall\, e \in E$

Основная проблема в существующих работах заключается в отсутствии интегрированного подхода к планированию, который одновременно учитывал бы выбор требований для проекта и возможности команды по их реализации в рамках ограничений ресурсов и времени. Традиционные методы не обеспечивают достаточной гибкости и адаптивности, необходимые для эффективного планирования в условиях неопределенности и динамично меняющихся требований как со стороны планирования задач, так и со стороны планирования расписаний.

Методы и материалы исследования

Для ускорения процесса поиска решения и повышения качества выполнения в NP-полных задач зачастую накладываются дополнительно ограничения планирования, в данном случае ограничения применимы следующие:

1. Количество рабочей силы ограничено.
2. Каждое требование имеет время окончания, необходимое время выполнения и издержки при задержке.
3. Проектов может быть более одного в одном makespan (количество времени, необходимое для обработки всех работ).
4. Параметры требований, сотрудников, команды могут изменяться в режиме реального времени.

5. Работы для обслуживания требований размещаются в доступные промежутки, стараясь минимизировать разрывы между этапами требований.

6. Если работа не может быть размещена для требований, требование полностью исключается из расписания.

7. Для каждого требования проверяется, все ли роли выполнены, все ли зависимости удовлетворены, не нарушен ли порядок.

8. Порядок ролей зафиксирован как дизайн → системный анализ → бэкенд / фронтенд разработка → тестирование. Данные роли являются стандартным набором ролей для самостоятельной кросс функциональной команды разработки программного обеспечения, так как охватывает весь жизненный цикл программного обеспечения от планирования и анализа требований до развертывания и обслуживания. Дизайн в этом случае необходим для создания прототипа интерфейса программного обеспечения, системный анализ для составления подробного технического задания для разработчиков, бэкенд разработка для реализации бизнес-логики и обеспечения хранения данных, фронтенд разработка для реализации интерфейса, а тестирование для контроля качества разработанного программного обеспечения перед его непосредственной реализацией.

Тогда формальную постановку задачи можно охарактеризовать так, пусть:

I — множество требований пронумерованных $1, \dots, |\mathrm{I}|$., каждое требование i состоит из набора ролей R_i. При этом между некоторыми парами требований также могут быть заданы ограничения предшествования: $i \to j$

R_i — множество ролей (работ) r внутри требования i. Требование считается выполненным, если по нему выполнены все работы для каждой имеющейся в требовании роли.

E — множество сотрудников e пронумерованных $1, \dots, |\mathrm{E}|$.. Каждый сотрудник обладает одной определённой ролью, а также имеет дни отпуска.

T — множество доступных дней t пронумерованных $1, \dots, |\mathrm{T}|$.

d_{ir} — длительность в днях работы роли r для обслуживания требования i. Оценка вычисляется через метод оценки и анализа проектов, а также с учётом производительности команды.

s_{ir} — время начала выполнения работы роли r для обслуживания требования i.

sb_{ir} — время начала выполнения работы роли r для обслуживания требования i в базовом расписании перед перепланированием.

$\Omega_e \subseteq 0, \dots, |\mathrm{T}|$ — множество дней, в которые сотрудник e недоступен по производственному календарю.

$\mathrm{r}_e \subseteq \mathrm{R}$ — множество ролей, которые может выполнять сотрудник e. Если каждый сотрудник специализируется на одной роли, то $|\mathrm{r}_e| = 1$.

U_i — «полезность» требования i.

Набор параметров штрафов:

P_1 за неполноценный требование, когда хотя бы одна роль не смогла быть запланирована,

P_2 за пересечение работ у одного сотрудника,

P_3 за нарушение технологической последовательности,

P_4 за выход за границы T.

$a_{\mathrm{ir}} \in \mathrm{E}$ — сотрудник, назначенный на роль r требования i.

$\mathrm{x}_{\mathrm{iert}} \in \{0,1\}$ — бинарная переменная, отображающая признак того, что работа роли r для требования i начинается в день t и выполняется сотрудником e, связанная с остальными переменными как:

$$\mathrm{x}_{\mathrm{iert}} = \begin{cases} 1, & \text{если } a_{\mathrm{er}} = e \text{ и } s_{\mathrm{ir}} = t \\ 0, & \text{иначе} \end{cases}.$$

Такая бинарная переменная позволяют удобнее формализовать ресурсные ограничения по дням и штрафы за пересечения.

$b_{\text{et}} \in \{0,1\}$ — бинарная переменная, отображающая признак того, что сотрудник *e* работал в день *t*, однако сотрудник может, например, не работать из-за назначенного отпуска, тогда для общего простоя отпускные дни не должны учитываться:

$$\text{b}_{et} = \begin{cases} 1, & \text{если сотрудник } e \text{ занят в день } t \text{ и } t \notin \Omega_i \\ 0, & \text{иначе} \end{cases}.$$

Ограничение на то, что каждая роль каждого требования выполняется один раз:

$$\sum_{e \in E} \sum_{t=0}^{T-d_{\text{ir}}} \text{x}_{\text{iert}} = 1 \text{ для всех } i \in I, r \in \text{R}_\text{i}.$$

Тогда штраф за неполноценно запланированное требование P_1 принимает вид:

$$P_1 = \sum_{i \in I} \sum_{r \in R_i} \mathbf{1}\left\{\sum_{e \in E} \sum_{t=0}^{T-d_{\text{ir}}} \text{x}_{\text{iert}} \neq 1\right\} * v_1.$$

Для каждой пары $(i,r), i \in I, r \in \text{R}_\text{i}$ должно существовать ровно одно значение s_{ir} и a_{ir} выбирается из тех $e \in E$, кто может выполнять роль *r*. Если роль *r* требует навыков, которые есть только у некоторых сотрудников, то:

$$\text{x}_{\text{iert}} = 0 \text{ для всех } i, t \text{ если } r \notin \text{R}_\text{i}$$

Аналогично a_{ir} не может принимать значение сотрудник e, если $r \notin \text{R}_\text{i}$. Если день $d \in \Omega_e$ выпадает на отпуск, сотрудник *e* не может выполнять работу:

$$\text{Если } \text{x}_{\text{iert}} = 1, \text{то } [t, t + d_{\text{ir}} - 1] \cap \Omega_e = \emptyset.$$

Штраф P_2 за пересечение работ у одного сотрудника. В любой день *t*, сотрудник *e* может выполнять не более одной роли, где интервал времени, когда сотрудник работал над обеспечением требования, рассчитывается как $t' : t' \leq t \leq t' + d_{\text{ir}}$ тогда штраф за день t вычисляется как:

$$P_2 t = \sum_{e \in E} \mathbf{1}\left\{\sum_{i \in I} \sum_{r \in R_i} \sum_{t' \in T} \text{x}_{\text{iert}'} > 1\right\} * v_2.$$

При этом общий штраф для всех пересекаемых работ:

$$P_2 = \sum_{t \in T} P_2 t.$$

Если роль r_2 не может начинаться до завершения r_1, например аналитика может начинаться только после дизайна, то:

$$s_{\text{ir2}} \geq \text{s}_{\text{ir1}} + \text{d}_{\text{ir1}}.$$

Некоторые работы ролей внутри одного требования могут идти параллельно как бэкенд и фронтенд разработка. Тогда они могут стартовать в любой момент после окончания предыдущего этапа:

$$s_{\mathrm{i},backend} \geq \mathrm{s}_{i,analytics} + \mathrm{d}_{\mathrm{i},analytics}, s_{\mathrm{i},frontend} \geq \mathrm{s}_{\mathrm{i},analytics} + \mathrm{d}_{\mathrm{i},analytics},$$

а штраф P_3 за нарушение технологической последовательности будет равен:

$$P_3 = \sum_{i \in I} \sum_{(r_1 r_2) \in i} \mathbf{1}\{s_{\mathrm{ir}2} < \mathrm{s}_{\mathrm{ir}1} + \mathrm{d}_{\mathrm{ir}1}\} * v_3.$$

Для каждой роли r требования i:

$$s_{\mathrm{ir}} + \mathrm{d}_{\mathrm{ir}} \leq T.$$

Если же работа выходит за границы проекта, то начисляется штраф P_4:

$$P_4 = \sum_{i \in I} \sum_{r \in R} \mathbf{1}\{s_{\mathrm{ir}} + \mathrm{d}_{\mathrm{ir}} > T\} * v_4.$$

Суммарное время простоя B рассчитывается как сумма всех дней простоев всех сотрудников:

$$B = \sum_{e \in E} \sum_{t=0}^{T} b_{\mathrm{et}}.$$

Общая полезность запланированных требований записывается как:

$$U = \sum_{i \in I} u_i .$$

Тогда целевая функция приспособленности, включающая несколько компонент:

$$F(x) = U - (P_1 + P_2 + P_3 + P_4) - B \rightarrow max.$$

Для динамического перепланирования рассчитывается, что, если в момент q появился новые требования или изменились ресурсы. Фиксируются уже запущенные работы, у которых $s_{ir} < q$. Такие работы не прерываются и продолжают выполняться по первоначальному плану. Для ролей, которые ещё не начаты к моменту q, разрешается изменить s_{ir} и/или назначить нового исполнителя. При этом, чтобы не нарушать первоначальный план, вводится стоимость корректировок K:

$$K = \sum_{i \in I} \sum_{r \in R} \mathbf{1}\{s_{ir} \geq q\} * v_k |s_{ir} - sb_{ir}|.$$

Итого необходимо минимизировать исходную функцию приспособленности с дополнительной стоимостью корректировок, причём

штрафы и полезности пересчитываются с учётом нового требований, новых отпусков и т.д. Таким образом, постановка динамического перепланирования может выглядеть так:

$$F(x) = \sum_{i \in I'} U' - (P_1 + P_2 + P_3 + P_4)' - B' - K \rightarrow max,$$

где I' является обновлённым множеством требований включая новый.

Приведённая формулировка полностью охватывает основные аспекты детального формирования расписания в условиях изменяющаяся среды с ограниченными ресурсами. Большая часть ограничений реализовано процедурно что значит осуществляется поиск допустимого окна для требования, назначение сотрудника с определенной ролью и проверку коллизий. Генетический алгоритм сводит всё к единому критерию [17, 18], где суммируются T, P, B и вычитается суммарная полезность. При динамическом перепланировании добавляется слагаемое стоимости корректировок относительно базового расписания.

Таким образом, данная модель является полным вариантом мультипроектной RCPSP с динамическим перепланированием, адаптированным к контексту требований, ролей-сотрудников и производственного календаря.

Результаты исследования

На основе реализованного алгоритма динамического перепланирования, который запускается при наличии изменений в глобальном календаре сотрудников или переоценке/выхода за оценку какой-либо из работ был разработан программный код с использованием генетического алгоритма. Основными понятиями генетического алгоритма является особь S_j закодированная с помощью набора генов и составляющая допустимое множество особей P, именуемой популяцией. Каждая особь имеет оценку приспособленности $F(x)$ как отражения качества особи, влияющая на вероятность выживания особи в текущем поколении. Для повышения

разнообразия популяции используются операторы скрещивания (создание новой особи путем объединения генетической информации родителей), мутация (изменение значений отдельных генов) и селекция (отбор наиболее приспособленных особей для скрещивания).

Для разработки которого необходимо представить динамичного перепланирования в виде пошагового алгоритма псевдокода:

Входные данные: Обновленные оценки работ d'_{ir}*, Текущее расписание* S*, Выполненные работы* Q*, Обновленный производственный календарь* T'
Выходные данные: Обновленное расписание S'
Начало

1. *Обновить длительность работ* $d_{ir} \leftarrow d'_{ir}$
2. *Обновить производственный календарь* Ω_e *для каждого* e *согласно* T'
3. *Определить фиксированные работы* $Q = \{i \in I \mid s_{ir} < q\}$
4. *Определить множество работ для перепланирования* $R = \{i \in I \cup Q\}$
5. *Проверить выполнимость исходного расписания* S *с обновленными* Ω_e *и* d_{ir}
6. *Если решение допустимо, то установить* $S' \leftarrow S$*, иначе перейти к шагу 7*
7. *Подготовить динамическое перепланирование:*
   a. *Определить работы для изменения* R'
   b. *Добавить функции штрафов допустимости* P_{1-4}
   c. *Добавить функцию корректирующего штрафа* K
8. *Применить генетический алгоритм*
   a. *Инициализировать популяцию* $P \leftarrow \{S_j\}$
   b. *Для популяции* $P = 1, \ldots, N$
      i. *Вычислить* $F(x)$ *для каждого* $S_j \in P$
      ii. *Применить селекцию, скрещивание и мутацию*
      iii. *Обновить популяцию* P
   c. *Выбрать лучшую особь* $S*$
9. *Сформировать итоговое расписание* $S' \leftarrow S * \cup Q$

Конец

Так, если работы с обновлёнными временными оценками могут быть вписаны в оставшееся расписание без изменения набора требований, алгоритм просто обновляет текущее расписание. В случае, если изменения по времени работ делают выполнение изначального плана невозможным, алгоритм запускает перепланирование, исключая требования, по которым уже начаты работы. Перепланирование основано на максимизации полезности, соблюдении зависимостей и оптимальном распределении оставшегося времени. После формирования нового расписания оставшиеся работы

интегрируются в существующий план, формируя итоговый оптимизированный график.

Для реализации проекта был выбран язык Python, поскольку он предоставляет широкие возможности для задач оптимизации, специализированных вычислений и визуализации данных, а также обеспечивает оперативность разработки. В качестве ключевых компонентов использовались библиотека NumPy для повышения скорости выполнения операций с векторами и матрицами, фреймворк DEAP для проведения эволюционных вычислений и библиотека Matplotlib, позволяющая визуализировать результаты в виде диаграммы Ганта.

При формировании расписаний оценка функции приспособленности проводилась с учетом таких критериев, как продолжительность выполнения работ, время простоя сотрудников, функции штрафов и суммарная полезность. Операторы кроссовера и мутации реализованы с использованием специализированных функций для работы со словарными структурами, отражающими распределение задач между ролями и сотрудниками. Кроме того, разработаны эвристические методы построения логики расписаний, в рамках которых для каждой работы, работы распределяются между сотрудниками с учетом доступного времени и этапов выполнения.

Особое внимание уделено гибкости настройки параметров алгоритма, а именно базовые структуры данных DEAP такие как типы индивидуумов, операторы кроссовера и мутации формируются на этапе инициализации основного кода, что значительно упрощает их параметризацию и модификацию. В качестве операторов кроссовера и мутации использовались функции cxTwoPointDict и mutShuffleDict, которые корректно работают со словарями, описывающими расписание, способствуя сохранению правильной структуры решений в процессе эволюции и повышая шансы на получение допустимого результата.

Использование NumPy позволяет существенно ускорить выполнение векторных операций, а модульная организация кода облегчает его профилирование и последующую оптимизацию. Выбор Python и фреймворка DEAP обеспечил высокую гибкость реализации, позволяя оперативно внедрять новые эвристики, корректировать параметры алгоритмов и адаптировать решения к требованиям задачи комплексного планирования и перепланирования расписания. Исходный код разработанного генетического алгоритма доступен на платформе GitHub по адресу: github.com/Flyneot/GenPlan.

Экспериментальная проверка

Для проведения эксперимента использовалась разработанная модель, дополненная параметрами, необходимыми для реализации генетического алгоритма. Исследование проводилось на выборке из 100 требований, разделенные на 500 работ, оптимальное распределение которых осуществлялось в рамках 60 рабочих дней. Команда проекта состояла из 10 сотрудников, каждый из которых выполнял одну из пяти ключевых ролей: разработчик фронтенд, разработчик бэкенд, системный аналитик, дизайнер или тестировщик. Некоторые работы имели взаимозависимости, при этом на одно требование приходилось не более пяти связей, что требовало обязательного учёта ограничений при оптимизации. Полезность каждого требования оценивалась на основе её приоритета и предполагаемого вклада в проект.

Генетический алгоритм функционировал с популяцией из 300 особей, каждая из которых представляла потенциальное решение. В процессе оптимизации применялись стандартные операции мутации (вероятность 0,1) и скрещивания (вероятность 0,9), а отбор для формирования следующего поколения осуществлялся по турнирному принципу с размером турнира равным пяти. На этапе формирования расписания анализировались параметры

распределения работ, такие как минимизация времени простоя сотрудников, равномерное распределение нагрузки между ролями и суммарная длительность выполнения работ. Для обеспечения воспроизводимости эксперимента каждая фаза повторялась 10 раз с последующим усреднением итоговых метрик.

Рис.1 иллюстрирует результаты формирования расписания.

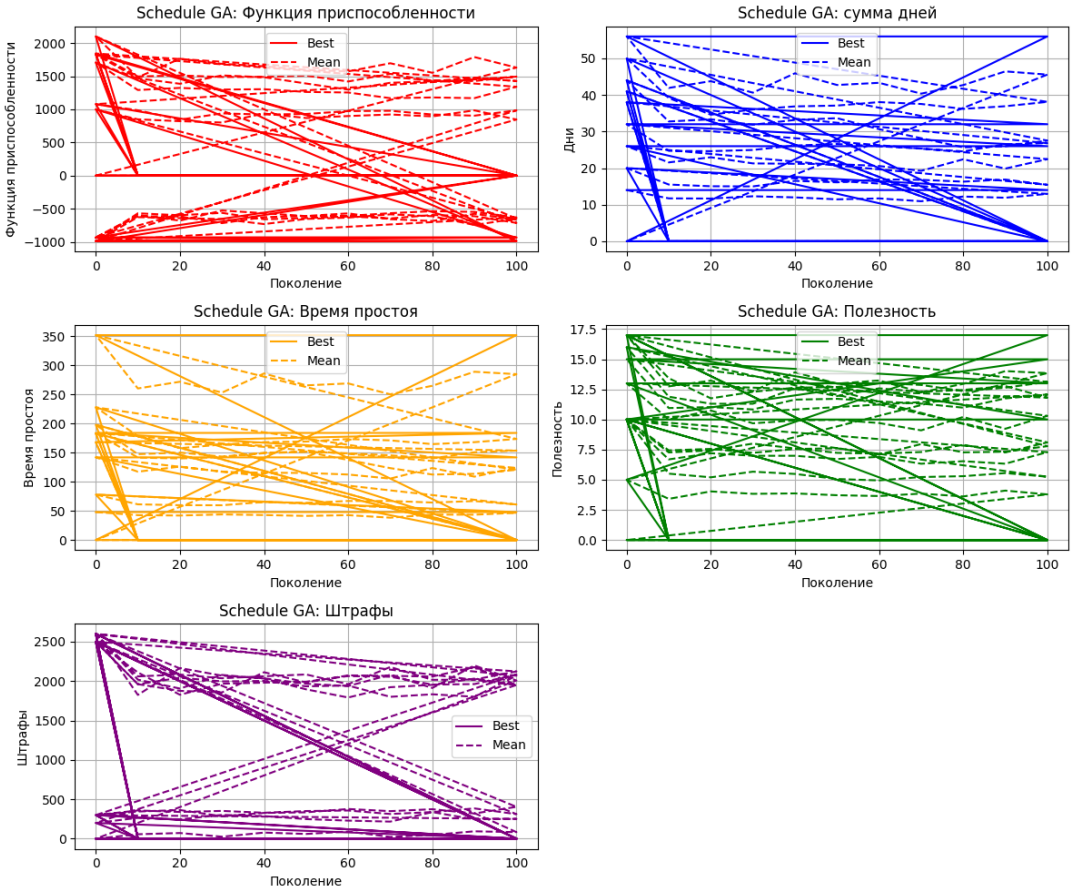

Рис. 1. – Характеристики работы генетического алгоритма планирования расписания

График распределения работ между сотрудниками демонстрирует, что алгоритм обеспечивает равномерное распределение нагрузки, минимизируя простои и гарантируя оптимальную загрузку каждого участника команды. При этом отмечаются незначительные пики нагрузки у дизайнеров и

разработчиков, что связано с высокой концентрацией работ в соответствующих направлениях, однако общее качество расписания остаётся на высоком уровне. Наличие нескольких значений одного и того же показателя на графиках вызвано тем, что в условиях мультипроектной среды для каждого проекта выполняется формирование расписания по отдельности, что существенно повышает скорость работы алгоритма, при это, ввиду наличия общего глобального календаря, не ухудшает результативность.

На рис.2 представлено итоговое плановое расписание, сформированное после нескольких итераций отбора требований и построения планов.

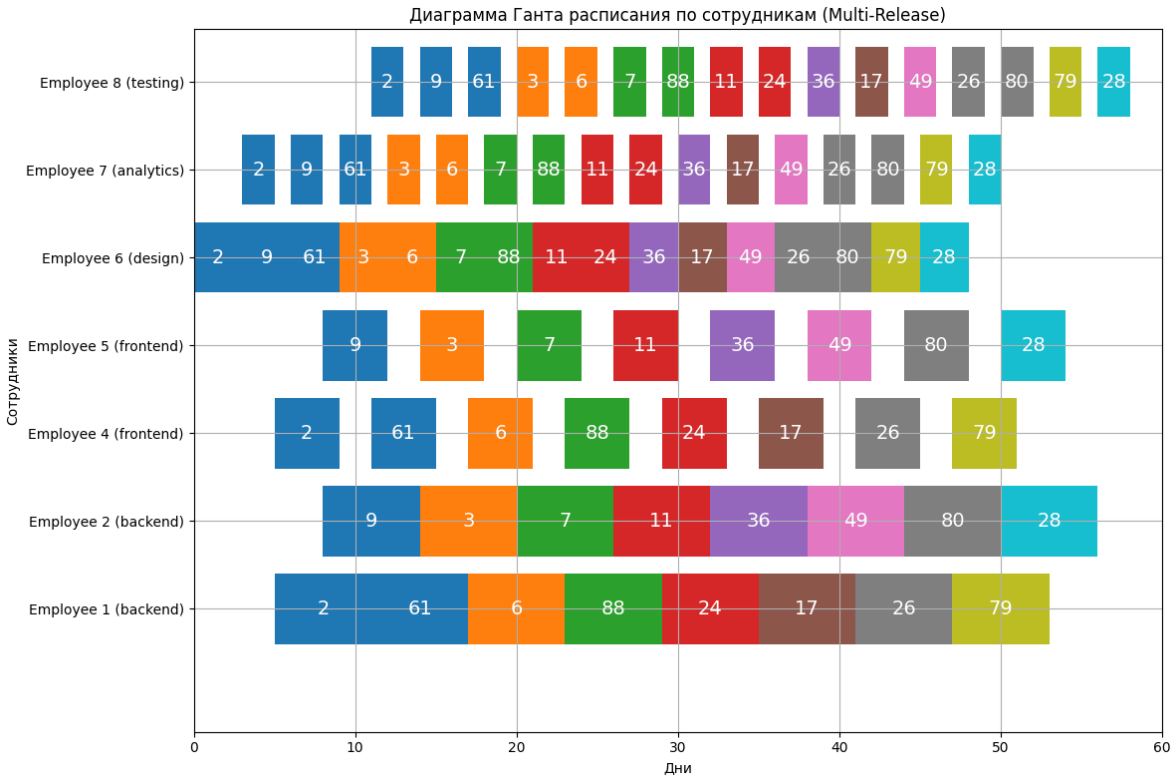

Рис. 2. – Диаграмма Ганта «График работ план»

По оси Y указаны сотрудники, сгруппированные по их ролям, а по оси X расположена временная шкала проекта. Каждый прямоугольник соответствует отдельной работе, расположенной по временной оси в

соответствии с её началом и длительностью, а по вертикали – согласно назначенному сотруднику. Диаграмма демонстрирует, что работы распределены между сотрудниками в зависимости от их специализации, а общая нагрузка сбалансирована, за исключением ролей дизайна и бэкенд разработки.

Также наблюдается, что система формирует осмысленное распределение работ, при котором сотрудники, исполняющие идентичные роли, нагружены заданиями различных требований в разные временные интервалы, что исключает продолжительные простои и параллельные конфликты. Итоговые метрики свидетельствуют о том, что при заданном наборе параметров и данных система генерирует решения, приближённые к оптимальному балансу между полезностью, риском, длительностью и эффективностью использования ресурсов.

Кроме того, для реализации механизма динамического перепланирования было учтено, что сотрудник №6 выходит в отпуск с 10 по 15 рабочий день. В результате система на 10-й день осуществила перепланирование, добавив два новых требования, не предполагающих участие дизайнера, и, ввиду их меньшей полезности по сравнению с полноценными требованиями, данные работы не были включены в первоначальный план (рис.3).

Таким образом, диаграмма демонстрирует, что требование под номером «28» было исключено из проекта, поскольку его выполнение выходило за рамки допустимого графика. Проверка разработанной системы на предмет эффективности отбора требований и составления для них расписания, с учётом заданных ресурсов и ограничений, проводилась путём анализа метрик полезности итогового набора требований, суммарных временных затрат на их реализацию, отсутствия значимых штрафов за нарушение последовательности

работ и выполнения зависимостей между ними, а также оптимальности распределения по сотрудникам.

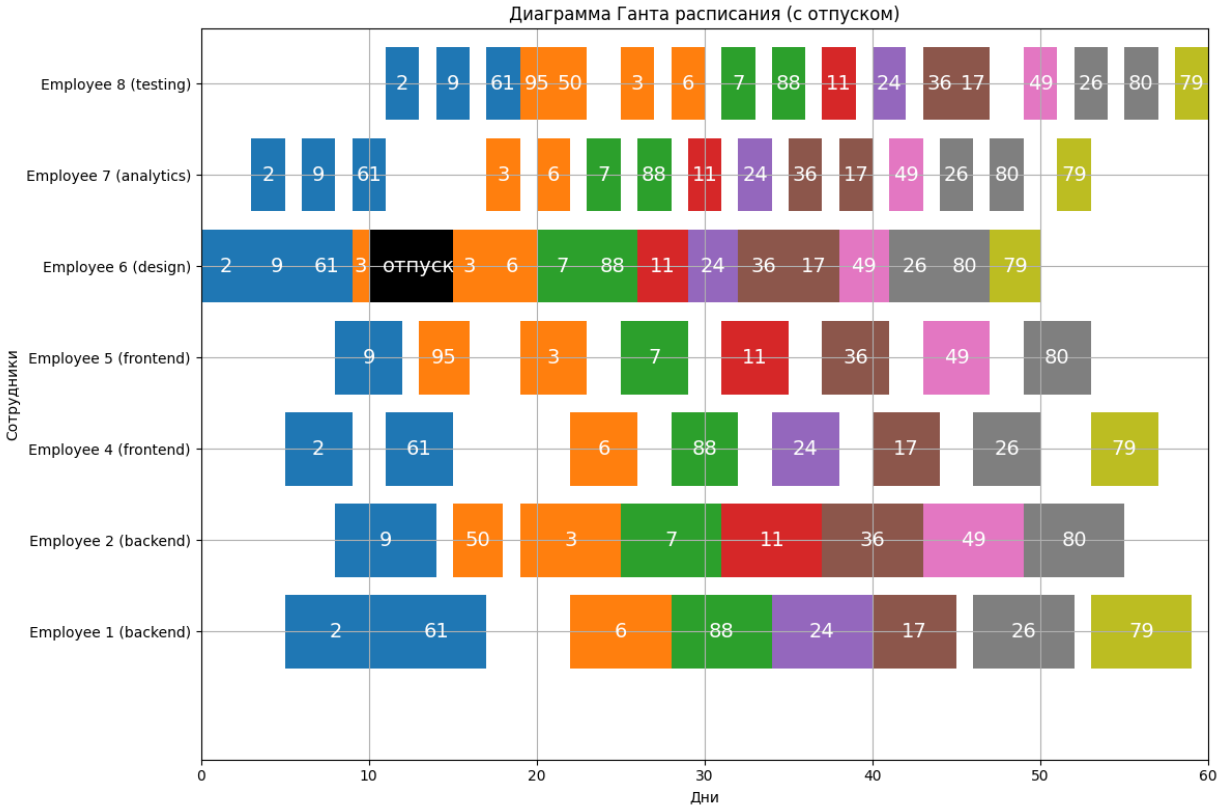

Рис. 3. – Диаграмма Ганта «График работ факт»

В итоге проведённые эксперименты подтвердили, что предложенный генетический алгоритм способен обеспечивать высокий уровень автоматизации планирования и перепланирования расписаний, что делает его перспективным для применения в реальных условиях управления проектами.

Заключение

В статье было описано решение задачи оптимизации процесса перепланирования проектов по разработке программного обеспечения отличающиеся тем, что, с целью обеспечение контроля за ходом спланированных работ, использует алгоритм динамичного перепланирования

расписания, а не фиксированное расписание от первоначального планирования. Было показано, что предложенный генетический алгоритм динамичного перепланирования способен обеспечить эффективное распределение работ даже при значительных изменениях исходных данных. Результаты эксперимента продемонстрировали высокую устойчивость модели, так как алгоритм успешно минимизирует простой сотрудников, оптимизирует загрузку и обеспечивает корректное выполнение зависимостей между требованиями. Итоговые метрики подтвердили, что использование гибких операторов кроссовера и мутации, а также интеграция дополнительных эвристических методов, позволяет достичь баланса между полезностью, риском и длительностью выполнения работ, что является важным показателем качества планирования в условиях ограниченных ресурсов.

На основе полученных результатов предлагается рассмотреть направления для дальнейших исследований. Одним из перспективных направлений является разработка гибридных моделей, объединяющих генетические алгоритмы с другими метаэвристическими методами, такими как алгоритмы частиц роя или муравьиной колонии, для более точного учета специфики многопроектной среды. Дополнительно, перспективно внедрение адаптивных механизмов корректировки параметров алгоритма в режиме реального времени, что позволит оперативно реагировать на изменения в условиях проекта и повышать качество планирования.

Таким образом, результаты исследования подтверждают потенциал использования динамичного перепланирования на основе генетических алгоритмов в управлении проектами по разработке программного обеспечения. Применение данной методологии способствует повышению гибкости и адаптивности процессов планирования, что особенно актуально в условиях быстро меняющихся требований. В дальнейшем интеграция алгоритмических решений в современные системы управления проектами

может обеспечить более эффективное распределение ресурсов и оптимизацию производственных процессов, что является важным шагом на пути к повышению конкурентоспособности в сфере разработки программного обеспечения.

Литература

1. Garey M.R., Johnson D.S., Sethi R. The complexity of flowshop and jobshop scheduling // Mathematics of Operations Research. 1976. V.1(2). pp. 117–120.

2. Manne A.S. On the Job-Shop Scheduling Problem // Operations Research. 1960. V.8(2). pp. 219–223.

3. Зак Ю. А. Двухстадийные задачи планирования для поточной линии // Проблемы управления. 2019. №6. С. 52-62.

4. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. Sequencing and Scheduling: Algorithms and Complexity // Handbooks in operations research and management science. 1993. V. 4. pp. 445-522.

5. Добрынин В. Н., Мороз В. В., Миловидова А. А. Унифицированная методика решения задачи расписания на основе задачи упорядочения //Системный анализ в науке и образовании. 2010. №3. С. 5-11.

6. Щербина О. А. Удовлетворение ограничений и программирование в ограничениях // Интеллектуальные системы. 2011. Т.15. №1-4. С. 53-170.

7. Yang B., Geunes J. Resource-Constrained Project Scheduling: Past Work and New Directions // Department of Industrial and Systems Engineering, University of Florida. Tech. Rep. 2001. 28 p.

8. Gómez Sánchez M., Lalla-Ruiz E., Fernández Gil A., Castro C., Voss S. Resource-Constrained Multi-Project Scheduling Problem: A Survey // European Journal of Operational Research. 2023. V. 309. № 3. pp. 958-976.

9. Habibi F., Barzinpour F., Sadjadi S. Resource-constrained project scheduling problem: review of past and recent developments // Journal of Project Management. 2018. V.3. pp. 55-88.

10. Roy B., Sen A. K. A novel metaheuristic approach for resource constrained project scheduling problem // Soft Computing: Theories and Applications: Proceedings of SoCTA 2019. Springer Singapore, 2020. pp. 535-544.

11. Coelho J., Vanhoucke M. Going to the core of hard resource-constrained project scheduling instances // Computer Operation Research. 2020. V.121. URL: doi.org/10.1016/j.cor.2020.104976.

12. Pérez E., Posada M., Lorenzana A. Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms // Soft Computing. 2016. V. 20. pp. 1879-1896.

13. Van Eynde R., Vanhoucke M. Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling // Journal of Scheduling. 2020. V. 23. pp. 301-325.

14. Wauters T., Kinable J., Smet P., Vancroonenburg W., Vanden Berghe G., Verstichel J. The multi-mode resource-constrained multi-project scheduling problem // Journal of Scheduling. 2016. V. 19. pp. 271-283.

15. Yu D., Lin J. Resource-Constrained Multi-Project Reactive Scheduling Problem With New Project Arrival // IEEE Access. 2023. V. 11. pp. 64370-64382.

16. Рябцев Т., Антонова Е. Модель интеллектуальной системы управления ненадежными элементами (людьми) // Proceedings of the XIIIth International Conference “KnowledgeDialogue Solution”, ITHEA, Bulgaria. 2007. Т. 1. С. 287-293.

17. Ляшов М.В., Береза А.Н., Коцюбинская С.А. Аппаратно-ориентированный генетический алгоритм синтеза конечных автоматов // Инженерный вестник Дона. 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5259.

18. Нетёсов А.С. Эволюционно-генетический подход к решению задач оптимизации. Сравнительный анализ генетических алгоритмов с

традиционными методами оптимизации // Инженерный вестник Дона. 2011, №3. URL: ivdon.ru/ru/magazine/archive/n3y2011/459.

References

1. Garey M.R., Johnson D.S., Sethi R. Mathematics of Operations Research. 1976. V.1(2). pp. 117–120.

2. Manne A.S. Operations Research. 1960. V.8(2). pp. 219–223.

3. Zak Yu. A. Problemy upravleniya. 2019. №6. pp. 52-62.

4. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. Handbooks in operations research and management science. 1993. V. 4. pp. 445-522.

5. Dobrynin V. N., Moroz V. V., Milovidova A. A. Sistemnyy analiz v nauke i obrazovanii. 2010. №3. pp. 5-11.

6. Shcherbina O. A. Intellektual'nye sistemy. 2011. V.15. №1-4. pp. 53-170.

7. Yang B., Geunes J. Department of Industrial and Systems Engineering, University of Florida. Tech. Rep. 2001. 28 p.

8. Gómez Sánchez M., Lalla-Ruiz E., Fernández Gil A., Castro C., Voss S. European Journal of Operational Research. 2023. V. 309. № 3. pp. 958-976.

9. Habibi F., Barzinpour F., Sadjadi S. Journal of Project Management. 2018. V.3. pp. 55-88.

10. Roy B., Sen A. K. Soft Computing: Theories and Applications: Proceedings of SoCTA 2019. Springer Singapore, 2020. pp. 535-544.

11. Coelho J., Vanhoucke M. Computer Operation Research. 2020. V.121. URL: doi.org/10.1016/j.cor.2020.104976.

12. Pérez E., Posada M., Lorenzana A. Soft Computing. 2016. V. 20. pp. 1879-1896.

13. Van Eynde R., Vanhoucke M. Journal of Scheduling. 2020. V. 23. pp. 301-325.

14. Wauters T., Kinable J., Smet P., Vancroonenburg W., Vanden Berghe G., Verstichel J. Journal of Scheduling. 2016. V. 19. pp. 271-283.

15. Yu D., Lin J. IEEE Access. 2023. V. 11. pp. 64370-64382.

16. Ryabtsev T., Antonova E. Proceedings of the XIIIth International Conference “KnowledgeDialogue Solution”, ITHEA, Bulgaria. 2007. V. 1. pp. 287-293.

17. Lyashov M.V., Bereza A.N., Kotsyubinskaya S.A. Inzhenernyj vestnik Dona. 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5259.

18. Netesov A.S. Inzhenernyj vestnik Dona. 2011, №3. URL: ivdon.ru/ru/magazine/archive/n3y2011/459.

Дата поступления: 3.02.2025

Дата публикации: 26.03.2025