Литература

1. Hogan, L.C. Designing for Performance: Weighing Aesthetics and Speed / L.C. Hogan. – O'REILLY, 2019. – 182 p.
2. Grigorik, I.K. High Performance Browser Networking: What Every Web Developer Should Know About Networking and Web Performance / I.K. Grigorik. – O'REILLY, 2018. – 398 p.
3. Microsoft Corporation. Performance Testing Guidance for Web Applications. – Microsoft Press, 2007. – 288 p.
4. Prihozhy, A.A. Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy. – Minsk: BNTU, 2019. – 229 p.
5. Прихожий, А.А. Модель и алгоритм оптимизации назначения объектов на узлы распределенной информационно-вычислительной системы / А.А. Прихожий // Доклады БГУИР. – 2010. – № 4. – С. 69 – 76.
6. Прихожий, А.А. Оптимизация размещения объектов с учетом их репликаций на узлах распределенной информационно-вычислительной системы / А.А. Прихожий // Информатика. – 2010. – № 3. – С. 124 – 134.

УДК 004.4 - 004.9

ПОИСК ОПТИМАЛЬНОГО НАЗНАЧЕНИЯ ПРОГРАММИСТОВ НА ГРУППЫ ИСХОДЯ ИЗ ТРЕБОВАНИЙ К НАВЫКАМ

Волоско А.Д.
Научный руководитель – Прихожий А.А., д.т.н., профессор

Для грамотной разработки приложений существует необходимость в декомпозиции основной задачи на обособленные подзадачи. Для этого надо решить две проблемы: разделить проект на взаимодействующие между собой части и как можно лучше распределить программистов на группы, для разработки данных частей. Если для решения первой проблемы достаточно хорошему специалисту проанализировать требования к программному продукту, то для решения второй необходимо учесть множество различных требований и использовать специальные методы [1, 2]. Данная статья представляет описание распределения программистов по группам с использованием генетического алгоритма, являющегося дальнейшим развитием алгоритма [3, 4]. Исходные данные для генетического алгоритма включают: технологии и их рейтинги, программисты и их квалификация в выбранных технологиях, спецификации групп программистов и требования к ним.

Технологии были разделены по назначению на 5 категорий. Для подсчёта рейтинга языков программирования используются результаты исследования ассоциации IEEE Spectrum [1, 5]. Для систем контроля версий был выставлен рейтинг 0,3, для баз данных, сред разработки и операционных систем – 0.5. Выбранные технологии представлены ниже на рисунке 1.

№	Название	Код	Тип	Рейтинг	№	Название	Код	Тип	Рейтинг
1	JavaScript	JS	Язык программирования	0,82	12	Jira	JR	Контроль версий	0,3
2	Python	Py	Язык программирования	1	13	Microsoft SQL	MS	Базы данных	0,5
3	Java	J	Язык программирования	0,89	14	PostgreSQL	P	Базы данных	0,5
4	PHP	PH	Язык программирования	0,52	15	Oracle	O	Базы данных	0,5
5	C#	CS	Язык программирования	0,76	16	MySQL	MY	Базы данных	0,5
6	C++	CPP	Язык программирования	0,72	17	Visual Studio	VS	Среда разработки	0,5
7	Html/CSS	HC	Язык программирования	0,49	18	Intellij Idea	II	Среда разработки	0,5
8	C	C	Язык программирования	0,65	19	PyCharm	PC	Среда разработки	0,5
9	SQL	SQL	Язык программирования	0,72	20	Eclips	EC	Среда разработки	0,5
10	Git	GT	Контроль версий	0,3	21	Windows	W	Опереационная система	0,5
11	Azure DevOps	AD	Контроль версий	0,3	22	Linux	L	Опереационная система	0,5

Рис. 1 – Выбранные технологии

Для оценки навыков программистов используется пятиразрядная градация [1] от 0 до 1, с промежутком 0.25. Где 0 – отсутствие навыков, 0.25 – минимальные, 0.5 – промежуточные, 0.75 – высокие, 1 – профессиональные навыки и знания в выбранной технологии. Для проведения исследования были случайным образом сгенерированы навыки 24 программистов. Сведения о навыках программистах представлены ниже на рисунке 2, где технологии представлены столбцами, квалификация программистов – строками, цифровая градация уровня программистов представлена столбиками: отсутствие столбика соответствует нулю, минимальный столбик – 0.25, средний – 0.5, выше среднего – 0.75 и максимальный – 1.

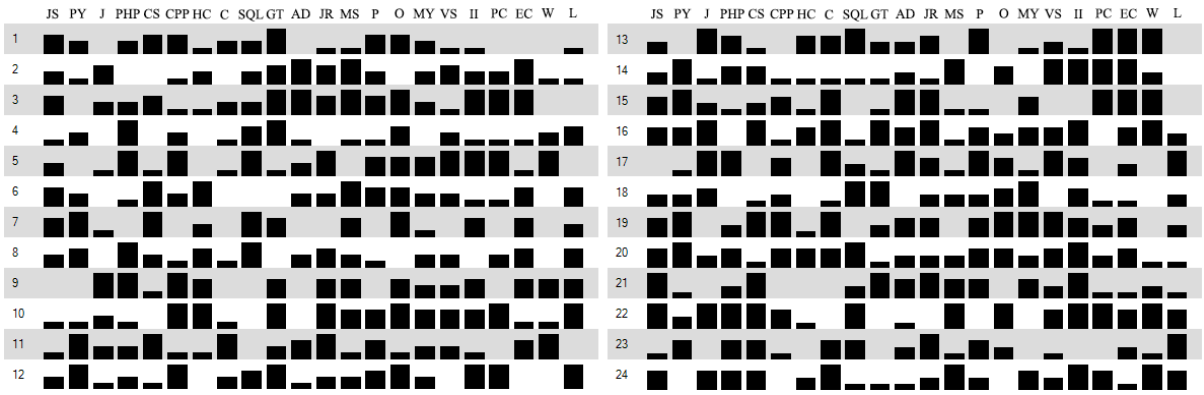

Рис. 2 – Навыки и знания программистов в выбранных технологиях

Целью явилось формирование 5 групп программистов названных «Site layout», «Frontend part», «Backend part», «Backend logic», «Database process». Помимо этих пяти групп программистов предусмотрена группа «Unemployment workers», которая включает неназначенных программистов в найденном оптимальном решении. Для каждой технологии, используемой каждой из групп, определены требования к уровню квалификации каждого и лучшего программиста. Требования к группам выглядят следующим образом: «Backend logic» {Java, 0.25, 0.75; Git, 0, 0.25; Intellij Idea 0.25, 0.5; Windows 0.25, 0.5}; «Site layout» {Html/CSS, 0.25, 0.75; Git, 0, 0.25; Intellij Idea, 0, 0.25}; «Frontend part» {JavaScript, 0.25, 0.5; Java, 0.25, 0.5; Html/CSS 0.25, 0.5; Git, 0, 0.25; Intellij Idea, 0, 0.25}; «Backend part» {Java, 0.25, 0.5; Git, 0, 0.25; Intellij Idea, 0.25, 0.5}; «Database process» {Java, 0.25, 0.5; SQL, 0.25, 0.5; Git, 0, 0.25; PostgreSQL, 0.25, 0.5; Visual Studio, 0, 0.25}.

В исследовании применен метод расчета функции эффективности назначения программистов на группы, предложенный в [1, 3, 4], но с учётом наличия разных требований для каждой из групп, а также наличия ограничения на минимальное и максимальное количество программистов в группе. Поэтому пороговая взвешенная квалификация группы $Qualif(g)$ будет дополнительно принимать значение 0 тогда, когда количество программистов в группе не соответствует требуемому. При проведении экспериментов приняты следующие параметры в модели функции эффективности: пороговое значение общей квалификации равно 0.7, важность квалификации по лучшим программистам относительно средней квалификации – 0.5, минимальное число программистов в группе – 3 и максимальное – 5.

В основе поиска лучшего назначения программистов лежит генетический алгоритм, в котором хромосома представлена массивом, размерность которого равна числу программистов, а *i*-ое значение в массиве указывает группу, на которую назначен *i*-ый программист [1, 3, 4]. Параметры генетического алгоритма: количество хромосом в популяции равно 100, количество скрещиваемых хромосом равно 50, вероятность мутации хромосомы 0.05. Особенности работы алгоритма:

1. Для выбора хромосом, к которым применяется операция скрещивания, используется метод рулетки.
2. Не выбранные хромосомы переходят в следующее поколение.
3. На выходе каждой операции скрещивания получаем 4 хромосомы (2 родительских и 2 дочерних), из которых в следующее поколение переходят 2 лучших представителя (1 родительская и 1 дочерняя), а 2 другие переходят в список *NSC* не выбранных хромосом.
4. Если в новой популяции есть хромосомы со значением функции эффективности равным 0, а список *NSC* содержит хромосомы со значением

функции большим 0, тогда происходит замена хромосом равных нулю, на лучшие хромосомы из *NSC*.

Графики, показывающие промежуточные и конечные результаты работы генетического алгоритма, представлены на рисунке 3.

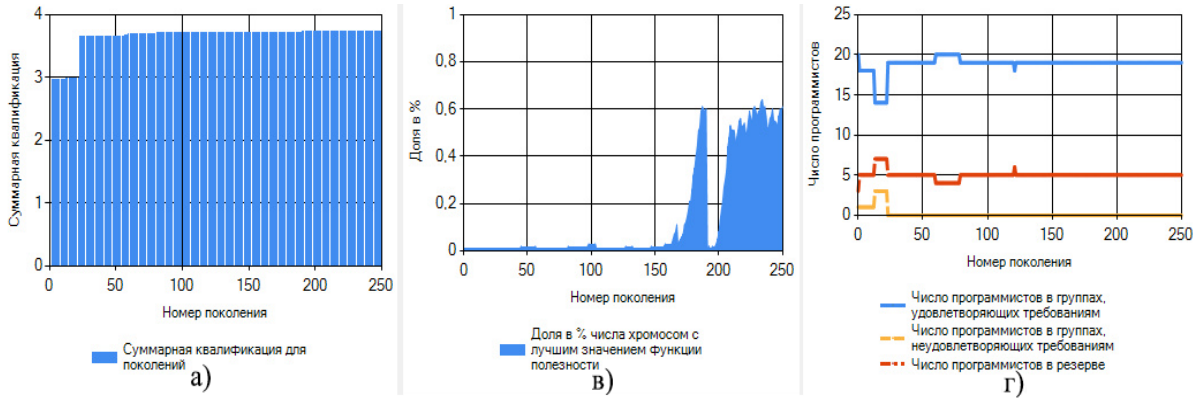

Рис.3 – Графики, показывающие процесс работы генетического алгоритма

В ходе исследования было найдено лучшее назначение программистов на группы, в котором значение функции эффективности равно 3.7557, а распределение программистов имеет вид: «Backend logic»: 13, 19, 24; «Backend part»: 3, 9, 11, 14, 18; «Database process»: 5, 6, 7, 17, 21; «Frontend part»: 10, 15, 22; «Site layout»: 12, 16, 20. Без назначения остались: 1, 2, 4, 8, 23.

Таким образом, в данной статье рассмотрен подход к распределению программистов по группам, имеющим разные требования. Подход может быть применен к решению проблемы распределения по группам не только программистов, но также и любых других работников, отвечающих за свою часть работы, в зависимости от навыков и специализации.

Литература

1. Прихожий, А.А. Метод оценки квалификации и оптимизация состава профессиональных групп программистов / А.А. Прихожий, А.М. Ждановский // Системный анализ и прикладная информатика. – 2018, №2. – С. 4 – 11.

2 .Prihozhy, A.A. Exact and greedy algorithms of allocating experts to maximum set of programmer teams / A.A. Prihozhy // System analysis and applied information science. – 2022, no. 1. – P. 40 – 46.

3. Prihozhy, A.A. Genetic algorithm of optimizing the qualification of programmer teams / A.A. Prihozhy, A.M. Zhdanouski // System analysis and applied information science. – 2020, no. 4, pp. 31 – 38.

4. Prihozhy, A.A., Zhdanouski, A.M. Genetic algorithm of optimizing the size, staff and number of professional teams of programmers / A.A. Prihozhy, A.M.

Zhdanouski // Open Semantic Technologies for Intelligent Systems, Minsk, BSUIR. – 2019. – P. 305 – 310.

5. Cass, S. Top Programming Languages 2022 [Электронный ресурс] / IEEE Spectrum, 2022 – Электрон. дан. – Режим доступа: https://spectrum.ieee.org/top-programming-languages-2022, свободный. – Дата доступа: 28.09.2022, – Загл. с экрана. – Яз. англ.

УДК 004.4

ОПТИМИЗАЦИЯ УЧЕБНОГО РАСПИСАНИЯ С ЦЕЛЬЮ УМЕНЬШЕНИЯ РАССТОЯНИЯ, ПРОХОДИМОГО СТУДЕНТАМИ МЕЖДУ КОРПУСАМИ

Домась А.А.
Научный руководитель – Прихожий А.А., д.т.н., профессор

Использование больших объемов информации, изменение условий образования, изменение средств и форм обучения, расширение спектра технических средств предполагают необходимость внедрения информационных технологий в образование [1 – 3].

Расписание учебных занятий является основой учебного процесса учебных заведений. То, насколько продуманно было составлено расписание, в конечном результате отражается на качестве управления учебным процессом. В процессе формирования расписания используется достаточно большой набор исходных данных, а также накладываются множества ограничивающих факторов, поэтому работа по составлению расписания требует значительных трудовых и временных затрат.

Задача составления расписания подразумевает распределение занятий таким образом, чтобы в одно и то же время не были назначены занятия у одной группы, одного преподавателя или в одной аудитории. Такая задача легко сводится к раскраске графа при условии, что известны все занятия для всех групп, которые надо провести, известны преподаватели, проводящие занятия, и известны аудитории, в которых занятия должны быть проведены. Одной из трудных проблем теории графов является проблема поиска хроматического числа, то есть поиска минимального числа цветов, необходимых для корректной раскраски вершин графа [4].

Для решения задачи составления учебного расписания, в данной работе использован алгоритм раскраски графа, построенный на жадной эвристике. В большинстве случаев жадные алгоритмы упорядочивают вершины по их степени (или относительной степени, то есть степени без учёта