УДК 004.051

Методика выбора компонентов стека технологий цифровых платформ на основе нечеткой логики

Д. Ю. Ильин[1]

Цифровые платформы с веб-интерфейсами становятся одним из распространенных видов информационных систем. Важной задачей дизайна и построения архитектуры программного обеспечения цифровой платформы является выбор используемых информационно-технологических решений и компонентов (стека технологий). В работе предложена методика выбора стека технологий на основе использования нечеткой логики. Методика на основе формализованного описания критериев оценки компонентов в заданных условиях функционирования цифровой платформы предполагает введение системы нечеткого вывода с последующим экспериментальным оцениванием критериев и направленным поиском эффективного решения на основе роевого интеллекта. Приведен пример использования методики для федеральной цифровой платформы массовых психологических исследований.

Ключевые слова: цифровые платформы, система нечеткого вывода, роевой интеллект, выбор программных компонентов.

1. Введение

Распространение веб-приложений, обусловленное их платформо- и аппаратно-независимостью, массовостью использования, широкими возможностями применяемых интерфейсов и протоколов обмена данными и программируемыми возможностями, определило развитие ИТ-направления – создание цифровых платформ [1]. Цифровизация различных сфер экономической и общественной жизни является одним из приоритетных направлений развития государства и мирового сообщества. Использование цифровых платформ позволяет осуществлять сбор информации и обмен ею между огромным количеством пользователей, объединяя результаты в большие данные [2]. Несмотря на обилие различных применений цифровых платформ, сформировался комплекс используемых в них современных информационных технологий (технологических решений), который принято называть «стек технологий» [3]. Важной особенностью элементов этого стека является их заменяемость [4], то есть одна из технологий может быть заменена на альтернативную или вновь созданную.

Широкое распространение для разработки цифровых платформ получило свободно распространяемое программное обеспечение (ПО) с открытым кодом, которое также способно решить все задачи реализации архитектуры вычислительной платформы, то есть существует большое количество свободного ПО стека технологий для реализации цифровых платформ [5, 6]. В этих условиях актуальной является задача выбора программно-технологических решений как при дизайне архитектуры, так и при ее обновлении или изменении условий функционирования используемой инфраструктуры вычислительного комплекса обработки данных. То есть для стека технологий задача интеграции программно-технологического реше-

[1] Работа выполнена при финансировании РФФИ, грант № 17-29-02198.

ния является задачей, требующей специальных теоретических концепций и практико-ориентированных методик.

Работа системы в целом зависит от эффективности каждого из компонентов стека технологий [7–13] и эффективности их взаимодействия [13]. При этом для одной задачи может существовать более чем одно готовое технологическое решение – как коммерческое, так и распространяемое бесплатно. На практике выбор происходит на основе результатов нагрузочных испытаний или экспертных оценок, данные подходы обобщают опыт использования конкретных компонентов или стека технологий, но не основываются на формальных оценках и не могут быть применены для сравнения эффективности. Существует множество решений задачи выбора эффективных программных компонентов [7–13] – разработаны методы решения задачи оптимизации, разработаны формальные модели задачи выбора. Однако они сводятся к решению задач идентификации и оптимизации большой размерности, предлагаемые методы не учитывают особенности эксплуатации, функционирования и инфраструктуры.

В работе предложен подход, основанный на нечеткой логике – инструменте, позволяющем как обобщить экспертный опыт, так и получить оценки и использовать формализованные методы поиска эффективных решений.

Нечеткая логика – это разновидность многозначной логики, в которой истинным значениям переменных соответствуют любые действительные числа в диапазоне (0; 1], что позволяет реализовать концепцию частичной истины. В противоположность булевой логике, согласно которой переменные могут принимать исключительно целочисленные значения 0 «ложь» и 1 «истина». Термин «нечеткая логика» стал широко использоваться в связи с теорией нечетких множеств, предложенной Лотфи Заде в 1965 году. Например, при выборе технологических решений цифровых платформ на основе минимизации потребления некоего ресурса определяющим является не конкретное количество байт или микросекунд, затраченных на исполнение алгоритма, изменяющееся незначительно от эксперимента к эксперименту, а качественная оценка, является ли потребление ресурсов «большим», «средним» или «малым» в соответствии с целями и представлениями разработчика. Введение таких качественных категорий позволяет существенно упростить оценку технологического решения, разбив все множество доступных технологических решений на небольшое количество классов относительно потребления ресурса, соответствующих качественным категориям.

Статья посвящена разработке программно-математических средства выбора компонентов стека технологий цифровых платформ на основе нечеткой логики.

2. Модель и методика выбора компонентов стека технологий

Модель выбора компонентов определена в соответствии с [14]. Пусть задано n функциональных требований q_i $(i=\overline{1,n})$ к цифровой платформе, а также t различных конфигураций ω^k $(k=\overline{1,t})$ информационной инфраструктуры, отражающих набор условий функционирования платформы. Разработчик платформы определяет M компонентов стека технологий для выбора. Каждый компонент из M реализует хотя бы одно из функциональных требований q_i. Подмножества альтернативных компонентов из M, способных реализовать функциональное требование q_i, обозначим m_i $(i=\overline{1,n})$. Подмножества компонентов, в которых для каждого функционального требования q_i существует хотя бы один компонент из M, определяются как стеки технологий s^j $(j=\overline{1,p})$. S – множество всех возможных стеков. Для оценивания интеграции программно-технологических решений в цифровые платформы вво-

дятся f показателей качества [13] $r_{\xi}^{k,j}$, $(\xi = \overline{1, f})$: $R^{k,j} = [r_1^{k,j}, \ldots, r_{\xi}^{k,j}, \ldots, r_f^{k,j}]^{\mathrm{T}}$, $(k = \overline{1, t}, j = \overline{1, p})$, принадлежащих пространству $\mathbb{R}^f$. Таким образом,

$$\forall \omega^k : s^j \to R^{k,j} \in \mathbb{R}^f,$$

где $R^{k,j}$ – вектор значения экспериментально вычисляемых частных показателей качества для конфигурации ω^k информационной инфраструктуры и оцениваемого стека s^j.

Предлагается следующая методика интегральной оценки качества стека технологий:

1. Формализация задачи выбора в соответствии с введенными выше определениями.
2. Формирование правил нечеткого вывода на основе целей и приоритетов разработчика цифровой платформы.
3. Исследование системы нечеткого вывода (СНВ) на полноту покрытия правилами диапазона входных значений, отсутствие избыточности правил, исключение ситуации неоднозначного выбора за счет настройки весов правил.
4. Выбор способа нормировки значений частных показателей $r_{\xi}^{k,j}$ для передачи на вход СНВ.
5. Организация экспериментов в виртуальной имитационной информационной инфраструктуре для получения нормированных значений $r_{\xi}^{k,j}$ для подачи на вход СНВ и получения с выхода СНВ интегрального показателя качества оцениваемого стека s^j для конфигурации ω^k виртуальной имитационной информационной инфраструктуры.
6. Для организации направленного поиска стека s^* для конфигурации ω^k используется алгоритм роевого интеллекта [15].

3. Результаты

Рассмотрим применение методики для выбора компонентов Node.js для разработки цифровой платформы DigitalPsyTools [16], предназначенной для информационной поддержки популяционных и лонгитюдных психологических исследований в России.

К компонентам архивации, используемым для передачи данных в цифровой платформе, предъявляются следующие функциональные требования: q_1 – «последовательно проверить все элементы массива на соответствие условию и вернуть массив, состоящий из элементов, для которых проверка дала значение «Истина», альтернативные компоненты: Lodash, Underscore; q_2 – «применить указанную функцию ко всем элементам массива, вернув новый массив, состоящий из преобразованных элементов», альтернативные компоненты: Lodash, Underscore, языковые средства JavaScript; q_3 – «вернуть первый элемент массива», альтернативные компоненты: Lodash, Underscore; q_4 – «сгенерировать полный путь к файлу или каталогу на основе указанного массива элементов пути», альтернативные компоненты: Path; q_5 – «найти и заменить подстроку в переданной строке», альтернативные компоненты: языковые средства JavaScript; q_6 – «выполнить архивирование переданного массива файлов и вернуть сгенерированный Zip-архив», альтернативные компоненты: Adm-zip, Jszip, Zipit; q_7 – «вычислить хеш MD5 для указанного набора данных», альтернативные компоненты: Hasha, md5, Ts-md5; q_8 – «считать данные из файла», альтернативные компоненты: Fs-Extra, Fs; q_9 – «считать содержимое каталога, вернув массив имен файлов и подкаталогов в каталоге», альтернативные компоненты: Fs-extra; q_{10} – «рекурсивно считать содержимое каталога и вернуть массив имен файлов и подкаталогов в каталоге», альтернативные компоненты: Recursive-readdir.

Таким образом, $n = 10, p = 216$.

Оценка качества функционирования производится относительно $f = 3$ частных показателей качества: $r_1^{k,j}$ – физическое время, затраченное на эксперимент, нс; $r_2^{k,j}$ – микропроцессорное время, потраченное на исполнение пользовательского кода во время эксперимента, мкс; $r_3^{k,j}$ – прирост размера страниц памяти, выделенных процессу эксперимента (включая кучу, кодовый сегмент и стек), байт. При проведении эксперимента частные показатели нормализованы относительно их максимальных значений в эксперименте, принимая значения на отрезке [0; 1].

Использование указанных показателей качества объясняется необходимостью выбора стека технологий, для которого потребление ресурсов в терминах прироста размера страниц памяти, затрат микропроцессорного и физического времени минимально, что позволит обеспечить наилучший пользовательский опыт на различных настольных и мобильных устройствах.

Для нечеткого вывода используется инженерный пакет прикладных программ Fuzzy Logic Toolbox for MATLAB. Он позволяет сделать процесс создания и настройки систем нечеткого вывода интерактивным. При этом разработчик визуально определяет количество нечетких множеств, вид функции принадлежности, метод фаззификации исходных количественных данных для перехода к качественному представлению, метод дефаззификации для получения количественного значения на выходе системы. Для решения используются следующие стандартные параметры нечеткого вывода Fuzzy Logic Toolbox: *and method: min*; *or method: max*; *implication: min*; *aggregation: max*; *defuzzification: centroid* [17].

Для получения интегральных оценок качества технологических решений $\Psi(\omega^k, s^j)$ были разработаны СНВ типа Мамдани (рис. 1), принимающие на вход три введенных выше частных показателя, при этом показатель $r_1^{k,j}$ обозначается как **t**, показатель $r_2^{k,j}$ обозначается как **cpu**, показатель $r_3^{k,j}$ обозначается как **ram**. Интегральный показатель $\Psi(\omega^k, s^j)$ обозначается как **quality**.

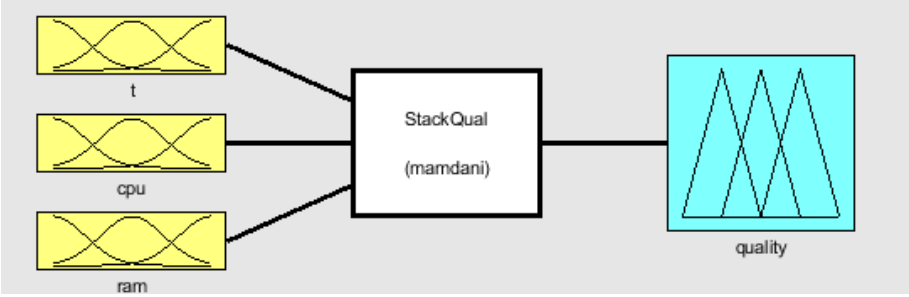


Рис. 1. Общая структура СНВ

В обеих СНВ численным значениям показателей t, cpu, ram ставятся в соответствие нечеткие множества “low” – низкое потребление ресурса, “med” – среднее потребление ресурса, “high” – высокое потребление ресурса, для которых заданы треугольные функции принадлежности с координатами вершин [-0,4 0 0,4], [0,1 0,5 0,9], [0,6 1 1,4] для “low”, “med”, “high” соответственно.

Интегральному показателю качества quality ставятся в соответствие нечеткие множества “low” – низкое качество, “med” – среднее качество, “high” – высокое качество в соответствии с определенными функциями принадлежности (рис. 2).

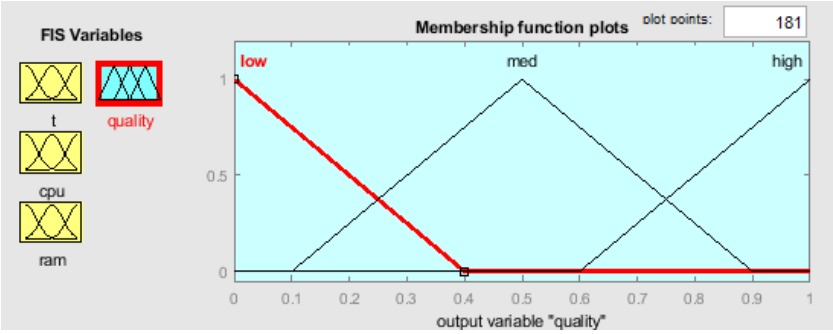


Рис. 2. Функции принадлежности нечетких множеств для интегрального показателя **quality**

При определении значения интегрального показателя **quality** в СНВ используются следующие правила (в скобках указан вес правила):

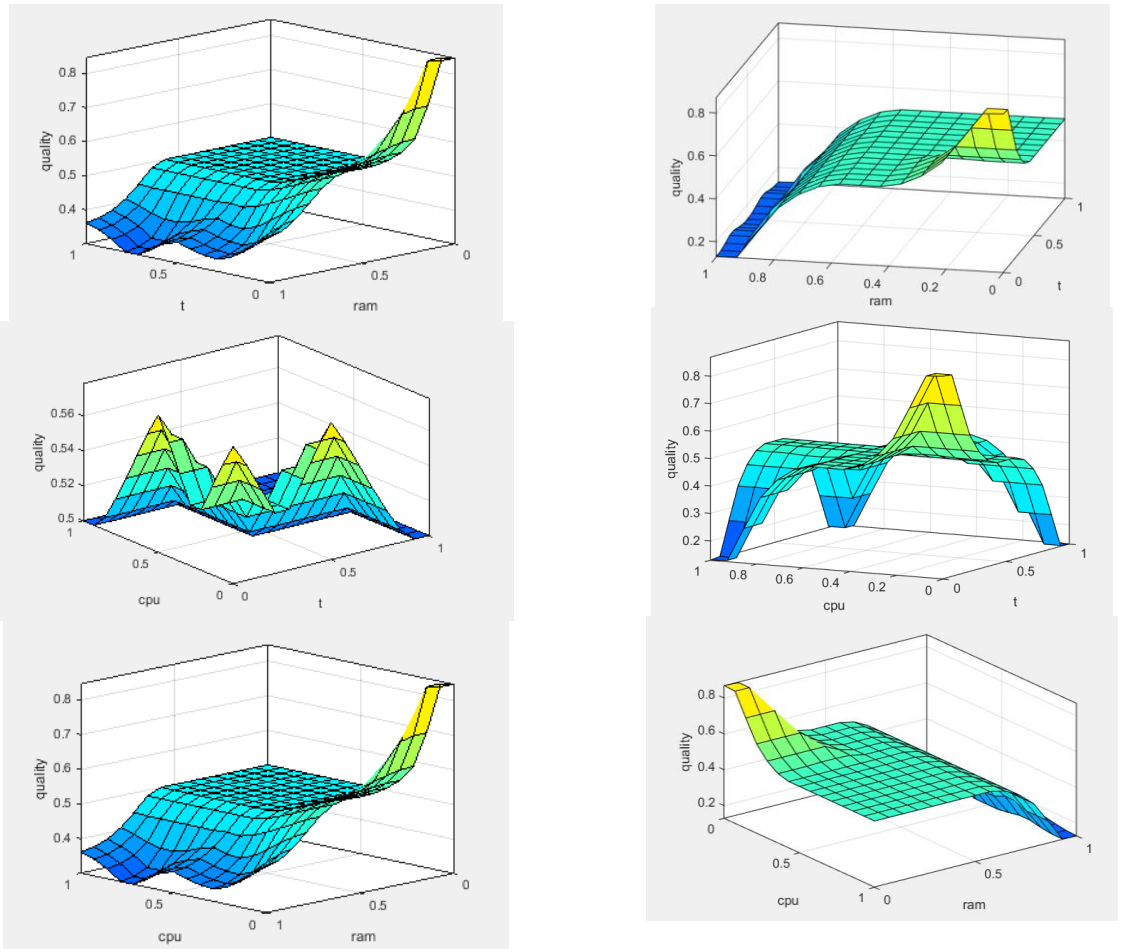
1. If (**t** is low) and (**cpu** is low) then (**quality** is high) (0.5)
2. If (**t** is low) and (**cpu** is med) then (**quality** is high) (0.5)
3. If (**t** is low) and (**cpu** is high) then (**quality** is med) (0.5)
4. If (**t** is med) and (**cpu** is low) then (**quality** is high) (0.5)
5. If (**t** is med) and (**cpu** is med) then (**quality** is med) (0.5)
6. If (**t** is med) and (**cpu** is high) then (**quality** is med) (0.5)
7. If (**t** is high) and (**cpu** is low) then (**quality** is med) (0.5)
8. If (**t** is high) and (**cpu** is med) then (**quality** is med) (0.5)
9. If (**t** is high) and (**cpu** is high) then (**quality** is med) (0.5)
10. If (**ram** is med) then (**quality** is med) (1)
11. If (**ram** is high) then (**quality** is low) (1)

Для проверки рассмотрен альтернативный вывод, использующий большее количество правил. При определении значения интегрального показателя quality в $СНВ_А$ используются следующие правила (в скобках указан вес правила):

1. If (**t** is low) and (**cpu** is low) and (**ram** is low) then (**quality** is high) (1)
2. If (**t** is low) and (**cpu** is low) and (**ram** is med) then (**quality** is high) (1)
3. If (**t** is low) and (**cpu** is low) and (**ram** is high) then (**quality** is med) (1)
4. If (**t** is low) and (**cpu** is med) and (**ram** is low) then (**quality** is high) (1)
5. If (**t** is low) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1)
6. If (**t** is low) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1)
7. If (**t** is low) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1)
8. If (**t** is low) and (**cpu** is high) and (**ram** is med) then (**quality** is low) (1)
9. If (**t** is low) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1)
10. If (**t** is med) and (**cpu** is low) and (**ram** is low) then (**quality** is high) (1)
11. If (**t** is med) and (**cpu** is low) and (**ram** is med) then (**quality** is med) (1)
12. If (**t** is med) and (**cpu** is low) and (**ram** is high) then (**quality** is low) (1)
13. If (**t** is med) and (**cpu** is med) and (**ram** is low) then (**quality** is med) (1)
14. If (**t** is med) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1)
15. If (**t** is med) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1)
16. If (**t** is med) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1)
17. If (**t** is med) and (**cpu** is high) and (**ram** is med) then (**quality** is med) (1)
18. If (**t** is med) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1)
19. If (**t** is high) and (**cpu** is low) and (**ram** is low) then (**quality** is med) (1)
20. If (**t** is high) and (**cpu** is low) and (**ram** is med) then (**quality** is low) (1)
21. If (**t** is high) and (**cpu** is low) and (**ram** is high) then (**quality** is low) (1)
22. If (**t** is high) and (**cpu** is med) and (**ram** is low) then (**quality** is med) (1)
23. If (**t** is high) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1)

24. If (**t** is high) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1)
25. If (**t** is high) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1)
26. If (**t** is high) and (**cpu** is high) and (**ram** is med) then (**quality** is low) (1)
27. If (**t** is high) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1)

Поверхности принятия решения СНВ и $СНВ_А$ показаны на рис. 3.



а) б)

Рис. 3. Поверхности принятия решения СНВ (а) и $СНВ_А$ (б)

Рассмотрение поверхностей принятия решения по парам показателей для СНВ и $СНВ_А$ указывает на применимость обеих СНВ для выбора технологических решений, однако за счет более компактной базы правил и использования весовых приоритетов СНВ отличается большей крутизной поверхности по показателям **t, cpu**, наличием локальных максимумов, что компенсируется превосходящим весом правил выбора по показателю **ram** (№ 10, № 11) для устранения неоднозначности выбора по **t, cpu**.

Для оценки значений критериев проведены экспериментальные исследования [14].

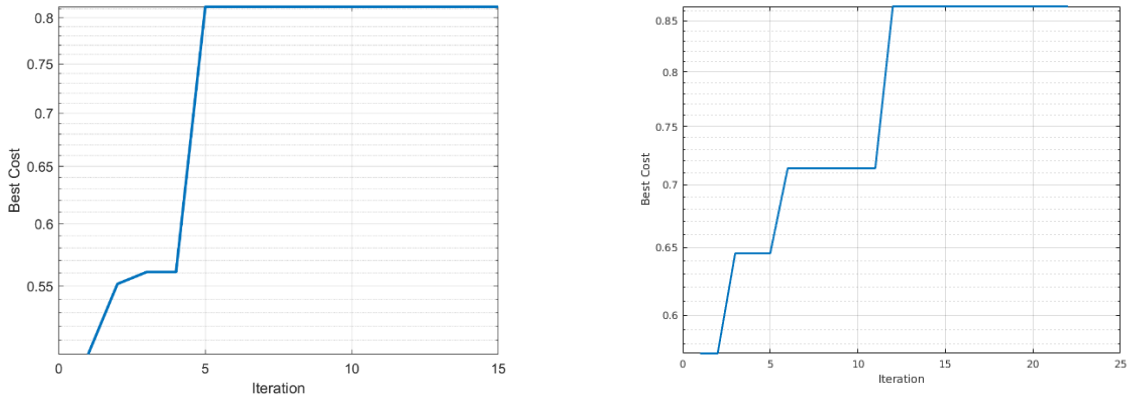
Для сокращения перебора и обеспечения направленного поиска s^* в пространстве стеков S использован роевой интеллект – алгоритм пчелиной колонии (АПК) [15]. АПК исполняется со следующими параметрами: размер колонии: 10; максимальное количество стагнирующих итераций: 10; порог сходимости: 0.0001; количество пчел-наблюдателей: 10; порог оставления источника: 20; коэффициент акселерации: 1.

При использовании СНВ АПК сошелся к решению задачи после 15 итераций. Координаты решения [2 3 1 1 1 1 1 2 1 1] соответствуют следующему выбору компонентов для реализации функциональных требований: q_1 реализуется компонентом Underscore; q_2 и q_5 реали-

зуются языковыми средствами JavaScript; q_3 реализуется компонентом Lodash; q_4 реализуется компонентом Path; q_6 реализуется компонентом Adm-zip; q_7 реализуется компонентом Hasha; q_8 реализуется компонентом Fs; q_9 реализуется компонентом Fs-extra; q_{10} реализуется компонентом Recursive-readdir. Выходное значение СНВ-1 для решения составило 0.8123.

При использовании $СНВ_А$ АПК сошелся к решению задачи после 22 итераций. Координаты решения [2 3 1 1 1 1 3 1 1 1] соответствуют следующему выбору компонентов для реализации функциональных требований к Платформе: q_1 реализуется компонентом Underscore; q_2 и q_5 реализуются языковыми средствами JavaScript; q_3 реализуется компонентом Lodash; q_4 реализуется компонентом Path; q_6 реализуется компонентом Adm-zip; q_7 реализуется компонентом Ts-md5; q_8 реализуется компонентом Fs-extra; q_9 реализуется компонентом Fs-extra; q_{10} реализуется компонентом Recursive-readdir. Выходное значение $СНВ_А$ для решения составило 0.8647.

Графики выбора компонентов под управлением АПК с использованием СНВ и $СНВ_А$ показаны на рис. 4.



а) б)

Рис. 4. График выбора компонентов под управлением АПК с использованием СНВ (а) и $СНВ_А$ (б)

Анализ результатов выбора стека технологий с использованием двух систем нечеткого вывода указывает на большую эффективность СНВ в сравнении с $СНВ_А$: за счет большей крутизны поверхностей СНВ по показателям t, cpu обеспечивается более быстрый выход к решению задачи выбора технологического решения. Разработка СНВ на основе компактной базы правил с различными весовыми коэффициентами также более проста, чем разработка $СНВ_А$.

Применение АПК к направленному поиску стека с использованием СНВ потребовало 312 экспериментальных оценок возможных стеков, прежде чем алгоритм сошелся. С использованием $СНВ_А$ потребовались 462 экспериментальные оценки. Использование полного перебора возможных стеков потребовало бы 1080 экспериментов в обоих случаях для выполнения хотя бы 5 оценок для расчета среднего для каждого стека из рассматриваемых 216. Оценка превосходящих подмножеств функциональных требований и альтернатив выбора компонентов полным перебором быстро привела бы к комбинаторному взрыву (рис. 5).

Таким образом, применение роевого интеллекта позволяет эффективно выбирать компоненты стека технологий на основе экспериментальных оценок за достаточно разумное время при приемлемых вычислительных усилиях, при этом неперспективные варианты выбора стека технологий автоматически отвергаются, в то время как оставшиеся для дальнейшего рассмотрения варианты подвергаются повторной оценке от итерации к итерации.

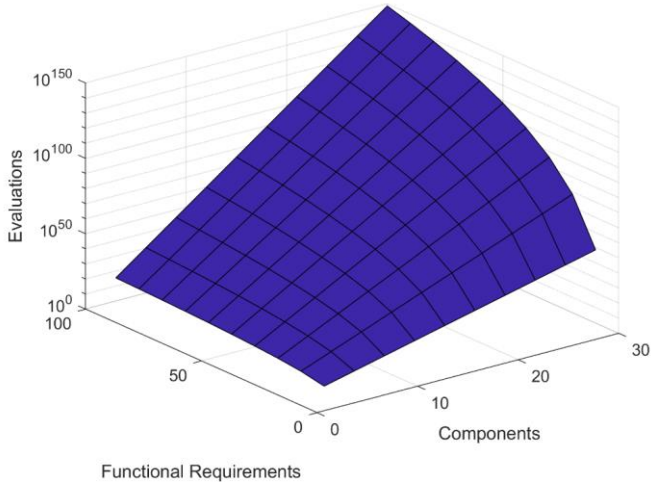


Рис. 5. Комбинаторный взрыв при полном переборе компонентов
(ось Z является логарифмической с основанием 10)

4. Заключение

Разработана методика выбора компонентов стека технологий цифровых платформ на основе нечеткой логики. Методика апробирована на выборе стека технологий для модуля архивации в цифровой платформе массовых психологических исследований. Полученные результаты подтверждены экспериментально, внедрение выбранных технологий обеспечило требуемый уровень качества и эффективности при сборе и передаче данных в массовых исследованиях [16].

В работе проведены исследования двух альтернативных методов нечеткого вывода, демонстрирующие применение нечеткой логики для разработанной методики. Для осуществления направленного поиска и сокращения перебора вариантов выбора стека технологий применен поиск решений на основе роевого алгоритма. Полученное решение позволило сократить количество итераций по оценке показателей в условиях заданной вычислительной инфраструктуры.

Предложенная методика может быть использована при выборе технологических решений в интегральной архитектуре современных цифровых платформ.

Литература

1. *Шабанов А. П.* Инновационное управление цифровыми платформами в экономике знаний // Системы управления, связи и безопасности. 2018. № 3. С. 106–135.
2. *Corbellini C. Mateos A. et al.* Persisting big-data: The NoSQL landscape // Information Systems. 2017. V. 63. P. 1–23.
3. *Семынин Д.* Стек для Больших Данных // Открытые системы. СУБД. 2014. № 1. С. 42–43.
4. *Brown E.* Web development with node and express: leveraging the JavaScript stack. O'Reilly Media, 2019.
5. *Лукинова О. В.* Вопросы проектирования цифровых платформ в парадигме открытых систем // Управление развитием крупномасштабных систем (MLSD'2018). М.: ИПУ РАН, 2018. С. 304–306.
6. *Комлева Н. В., Вилявин Д. А.* Цифровая платформа для создания персонализированных адаптивных онлайн курсов // Открытое образование. 2020. № 24 (2). С. 65–72.

7. *Beran P. P., Vinek E., Schikuta E.* A cloud-based framework for QoS-aware service selection optimization // Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, 2011. P. 284–287.
8. *Vescan A., Grosan C., Pop H. F.* Evolutionary algorithms for the component selection problem // 2008 19th International Workshop on Database and Expert Systems Applications. IEEE. 2008. P. 509–513.
9. *Ramírez A., Parejo J. A., Romero J. R., Segura S., Ruiz-Cortés A.* Evolutionary composition of QoS-aware web services: a many-objective perspective // Expert Systems with Applications. 2017. V. 72. P. 357–370.
10. *Ramirez A., Romero, J. R., Ventura S.* Interactive multi-objective evolutionary optimization of software architectures // Information Sciences. 2018. V. 463. P. 92–109.
11. *Yang Y., Yang B., Wang S., Jin T., Li S.* An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing // Applied Soft Computing. 2020. V. 87.
12. *Gholamshahi S., Hasheminejad S. M. H.* Software component identification and selection: A research review // Software Practice and Experience. 2019. V. 49, № 1. P. 40–69.
13. *Гусев А. А.* Поиск эффективного набора взаимодействующих компонентов программных систем на основе роевого интеллекта // Cloud of Science. 2019. Т. 6, № 3. С. 475–487.
14. *Ilin D., Gusev A., Nikulchev E.* Software Design using Genetic Quality Components Search // International Journal of Advanced Computer Science and Applications. 2019. V. 10, № 12. P. 48–54.
15. *Lun L., Chi X., Xu H.* Coverage criteria for component path-oriented in software architecture // Engineering Letters. 2019. V. 27, № 1. P. 40–52.
16. *Никульчев Е. В., Ильин Д. Ю. и др.* Разработка открытой цифровой платформы масштабных психологических исследований // Вестник РФФИ. 2019. № 4. С. 105–119.
17. Fuzzy logic toolbox user's guide. // Mathworks Inc., 2010

Статья поступила в редакцию 07.05.2020;
переработанный вариант – 11.06.2020.

Ильин Дмитрий Юрьевич
аспирант кафедры управления и моделирования систем МИРЭА – Российского технологического университета (119454 Москва, пр. Вернадского, 78); главный аналитик Дата Центра Российской академии образования, e-mail: i@dmitryilin.com.

Software component selection methodology based on fuzzy logic for digital platform design

D. Ilin

Digital platforms with web interfaces become one of the common types of information systems. An important task of the design and construction of the digital platform software architecture is the selection of the appropriate technological solutions and software components (technology stack). The paper proposes a methodology for selecting a technology stack based on the use of fuzzy logic. The methodology is based on the formalized criteria for evaluating the components in the given conditions of the functioning of the digital platform. It involves fuzzy inference system with subsequent experiment-based evaluation of the criteria and the directed search for an effective solution with the swarm intelligence algorithm. The paper also presents an example of the proposed methodology application for the design of the federal digital platform for mass psychological research.

Keywords: digital platforms, fuzzy logic, swarm intelligence, software design.