ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 69 за 2024 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

В.А. Судаков, А.Д. Шаблий

Оптимизация послепродажного обслуживания программного обеспечения в плагинной архитектуре

Рекомендуемая форма библиографической ссылки: Судаков В.А., Шаблий А.Д. Оптимизация послепродажного обслуживания программного обеспечения в плагинной архитектуре // Препринты ИПМ им. М.В.Келдыша. 2024. № 69. 17 с. https://doi.org/10.20948/prepr-2024-69
https://library.keldysh.ru/preprint.asp?id=2024-69

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

В.А. Судаков, А.Д. Шаблий

Оптимизация послепродажного обслуживания программного обеспечения в плагинной архитектуре

Москва — 2024

В.А. Судаков, А.Д. Шаблий

Оптимизация послепродажного обслуживания программного обеспечения в плагинной архитектуре

Оптимизация стоимостных издержек является актуальной проблемой в современных задачах разработки и послепродажного обслуживания программного обеспечения. В работе рассмотрена оптимизация стоимости послепродажного обслуживания плагинной архитектуры. Решена задача оптимальной декомпозиции функционала при условии его поставки в разных комплектациях. Разработана математическая модель, которая обеспечивает оптимизацию распределения файлов исходного кода по плагинам по критерию минимальной стоимости послепродажного обслуживания реализуемых ими требований. Предложена программная реализация модели на языке программирования Python с задействованием модуля Pyomo. Выполнена серия вычислительных экспериментов по решению задачи оптимизации различными решателями с целью поиска оптимальных значений переменных и оценки времени работы.

Ключевые слова: плагин, оптимизация, целевая функция, ограничения, решатели, Pyomo.

Sudakov Vladimir Anatolyevich, Shabliy Alexey Denisovich

Optimizing After-Sales Software Maintenance in Plugin Architecture

Cost optimization is a pressing issue in modern software development and after-sales service. The paper considers optimization of the cost of after-sales service for a plug-in system. The problem of optimal decomposition of functionality is solved under the condition of its delivery in different configurations. A mathematical model is developed that optimizes the distribution of source code files among plug-ins based on the criterion of the minimum cost of after-sales service for the requirements they implement. A software implementation of the model in the Python programming language using the Pyomo module is proposed. A series of computational experiments are performed to solve the optimization problem using various solvers in order to find optimal parameter values and estimate the operating time.

Key words: plugin, optimization, objective function, constraints, solvers, Pyomo.

Введение

В сфере информационных технологий на сегодняшний день актуально определение рациональной архитектуры программного обеспечения. Проанализировав работы [1], [2] и [3], можно сделать вывод, что оптимизировать необходимо не только техническую реализацию программных систем [4, 5], но и стоимость решения как программного продукта.

Анализ показал, что стоимость решения может зависеть от доступности и стоимости применяемых технологий [3], характера [1] реализации программного обеспечения (ПО), а также от требований к квалификации пользователей ПО [2]. Так, например, дополняющий друг друга функционал может снижать стоимость решения [4]. Повышать стоимость могут, например, конкурирующие за разделяемый ресурс единицы функционала [5].

Снижение стоимости может быть достигнуто за счет оптимизации расходов на послепродажное обслуживание (ППО). В рамках оптимизации из поставки решения исключается бесполезный для заказчика функционал. Такой способ был бы актуален для решений, которые динамически формируют конечный функционал программного комплекса, вносят в него новый и изменяют существующий. Примером таких решений являются плагинные системы [6, 7]. В них интеграционными функциональными единицами являются плагины [4, 6, 7]. Такой подход близок идеологии расширяемого ПО, развиваемой в ИПМ им. М.В.Келдыша РАН [8].

При создании ПО необходимо учесть:

- требования – в них описываются функциональные возможности ПО;
- файлы исходного кода – в них на языке программирования реализованы требования;
- плагины – они включают в себя файлы исходного кода;
- трассируемость требований на файлы исходного кода;
- зависимости между файлами исходного кода;
- распределение файлов исходного кода по плагинам.

Благодаря этим сведениям можно построить математическую модель, отвечающую на вопрос, какие требования будут реализованы в поставке ПО конкретному заказчику. Если имеющуюся информацию дополнить сведениями о стоимости ППО требований и механизмами формирования результирующей стоимости, тогда может быть решена задача оптимизации внутренней структуры программного решения по критерию минимальной стоимости ППО.

Описание модели

Пусть имеется n требований к ПО, которые трассируются на m файлов исходного кода. Файлы распределены по k плагинам. Требуется определить оптимальное распределение файлов по плагинам для минимальной стоимости ППО возможных поставок в заявленных l комплектациях. Стоимость ППО

требования в рамках поставки зависит от состава поставки и может изменяться при наличии реализованных в поставке других требований.

Решение, выполненное в виде комплекса плагинов, может быть поставлено более чем в одной комплектации. В рамках каждой из заявленных комплектаций каждое из возможных требований маркируется либо как полезное, либо как бесполезное. Поставка должна включать все полезные требования. Присвоенные каждому из требований признаки полезности во всех заявленных комплектациях образуют матрицу бинарных отношений $E_{l\times n} = \|e_{ij}\|$. Элемент $e_{ij} = 0$, если в рамках i-й комплектации j-е требование бесполезно, и 1, если полезно. Обозначим вектор-строку требований для одной комплектации i:

$$E_i = (e_{i1}, e_{i2}, e_{i3}, \dots, e_{in}).$$

Стоимость ППО каждого отдельного требования может отличаться в разных поставках. Это происходит из-за того, что по условию задачи наличие других реализованных требований в поставке изменяет стоимость ППО требования и может его как уменьшать, так и увеличивать. Образуется матрица изменения стоимости ППО требований $C_{n\times n} = \|c_{ij}\|$. В ней указывается, на сколько изменится стоимость ППО i-го требования, если в поставке будет реализовано j-е.

Каждое требование трассируется на файлы исходного кода. Для реализации требования в поставке в нее должны быть включены все файлы, которые реализуют данное требование. При этом каждое из требований может быть реализовано в одном или нескольких файлах, а каждый файл может реализовывать одно или несколько требований. Так образуется матрица трассируемости $T_{n\times m} = \|t_{ij}\|$. Каждый из элементов матрицы имеет значение в диапазоне $[0, 1]$. Значение определяет условную долю участия файла в реализации требования. Если j-й файл не задействован в реализации i-го требования, то значение $t_{ij} = 0$, иначе $0 < t_{ij} \leq 1$. При этом выполняются условия:

$$\sum_{j=1}^{m} t_{ij} = 1\,, \qquad i = \overline{1,n}.$$

Файлы исходного кода могут иметь зависимости друг на друга. Зависимости между ними описывает матрица бинарных отношений $D_{m\times m} = \|d_{ij}\|$. В ней элемент $d_{ij} = 0$, если у i-го файла нет зависимости от j-го, и равен 1, если зависимость есть. Значения элементов на главной диагонали $d_{ii} = 0$.

Файлы исходного кода распределены по плагинам. Один файл может быть включен только в один плагин, в то время как один плагин может включать в себя множество файлов. За это отвечает матрица бинарных отношений $A_{m\times k} = \|a_{ij}\|$. Элемент $a_{ij} = 0$, если i-й файл не включен в j-й

плагин, и равен 1, если включен. Кроме того, на элементы матрицы A действуют ограничения:

$$\sum_{j=1}^{k} a_{ij} = 1\,, \qquad i = \overline{1,m}.$$

Решение задачи поиска оптимальной декомпозиции заключается в нахождении значений оптимизационных переменных a_{ij}.

Целевая функция может быть записана как минимизация стоимостей поставок во всех возможных комплектациях:

$$\min_{A} \sum_{i=1}^{l} f_c(E_i, A, T, D).$$

где f_c – функция стоимости ППО комплектации.

Алгоритм вычисления значения функции $f_c(E_i, A, T, D)$ для одной заданной поставки i следующий:

1. Вычислить вектор полезных файлов исходного кода:
$$F_{1\times m} \leftarrow E_i \cdot T.$$
2. Определить минимально необходимый перечень файлов исходного кода для осуществления поставки в заданной комплектации. Для этого вычислить сумму F и разрешенных зависимостей:
$$\hat{F}_{1\times m} \leftarrow F + \sum_{j=1}^{m} \left(F \cdot D^{j}\right).$$
3. Определить плагины, которые должны войти в поставку:
$$P_{1\times k} \leftarrow f_{in}\left(\hat{F} \cdot A\right),$$
где f_{in} применяется к каждому элементу матрицы $\hat{F} \cdot A$ по следующему правилу:
$$f_{in}(x) = \begin{cases} 0 & \text{если } x = 0, \\ 1 & \text{если } x > 0. \end{cases}$$
4. Определить все файлы исходного кода, которые должны войти в поставку:
$$\dot{F}_{m\times 1} \leftarrow A \cdot P^{T}.$$
5. Определить все реализованные требования в поставке:
$$\dot{R}_{n\times 1} \leftarrow f_{im}\left(T \cdot \dot{F}\right),$$
где f_{im} применяется к каждому элементу матрицы $T \cdot \dot{F}$ по следующему правилу:
$$f_{im}(x) = \begin{cases} 0 & \text{если } x < 1, \\ 1 & \text{если } x \geq 1. \end{cases}$$
6. Рассчитать стоимость ППО комплектации:
$$f_c(E_i, A, T, D) \leftarrow \dot{R}^{T} \cdot C \cdot \dot{R}.$$

Выполнив все шаги алгоритма, получим следующую запись целевой функции:

$$\min_A \sum_{i=1}^{l} \left[f_{im} \left(T \cdot \left(A \cdot \left[f_{in} \left(\left(\left(E_i \cdot T + E_i \cdot T \cdot \sum_{j=1}^{m} D^j \right) \cdot A \right) \right]^T \right) \right) \right]^T \cdot C \cdot \right.$$
$$\left. \cdot f_{im} \left(T \cdot \left(A \cdot \left[f_{in} \left(\left(\left(E_i \cdot T + E_i \cdot T \cdot \sum_{j=1}^{m} D^j \right) \cdot A \right) \right]^T \right) \right) \right) \right).$$

Появление в целевой функции выражений, в которых оптимизационные переменные перемножаются между собой, делает модель нелинейной. Кроме того, нелинейными являются функции f_{im} и f_{in}. Нелинейность модели приводит к значительному усложнению поиска значений ее переменных и невозможности применения для этого большинства программных решателей, ориентированных на нахождение точного решения. Однако, если перемножаемые переменные являются бинарными, то модель может быть приведена к линейному виду. Выражение вида $x \cdot y$, где x и y – бинарные переменные, может быть заменено путем добавления дополнительной бинарной переменной $f = x \cdot y$. Для этого вводятся дополнительные ограничения:

$$\begin{cases} x + y \ \leq f + 1, \\ f \leq x, \\ f \leq y. \end{cases}$$

Доказательство корректности ограничений методом разбора случаев приведено в таблице 1. Из нее видно, что $f = x \cdot y$ тогда и только тогда, когда выполняются все три ограничения.

Таблица 1

x	y	f	$f_1 = (x + y \ \leq f + 1)$	$f_2 = (f \leq x)$	$f_3 = (f \leq y)$	$f_1 \wedge f_2 \wedge f_3$
0	0	0	true	true	true	true
0	0	1	true	false	false	false
0	1	0	true	true	true	true
0	1	1	true	false	true	false
1	0	0	true	true	true	true
1	0	1	true	true	false	false
1	1	0	false	true	true	false
1	1	1	true	true	true	true

Функция определения вхождения f_{in} применяется с целью преобразования большего или равного 0 входного параметра в бинарное значение.

Функцию f_{in} следует представить в виде переменных и ограничений в модели. Для этого используется метод big M [10]. Этот метод предписывает

заведение в модели бинарной переменной $f = f_{in}(x)$ и дополнение модели ограничениями:

$$\begin{cases} f < x + 1, \\ x \le M \cdot f. \end{cases}$$

Здесь M – условно большое число. Доказательство корректности ограничений методом разбора случаев приведено в таблице 2. Из нее видно, что $f = f_{in}(x)$ тогда и только тогда, когда выполняются оба ограничения.

Таблица 2

x	f	$f_1 = (f < x + 1)$	$f_2 = (x \le M \cdot f)$	$f_1 \wedge f_2$
0	0	true	true	true
0	1	false	true	false
(0;1)	0	true	false	false
(0;1)	1	true	true	true
1	0	true	false	false
1	1	true	true	true
(1; ∞)	0	false	false	false
(1; ∞)	1	true	true	true

Функция f_{im} применяется с целью преобразования большего или равного 0 входного параметра в бинарное значение. Функцию f_{im} следует представить в виде переменной f и ограничений $f = f_{im}(x)$. Для этого используется ранее упомянутый метод big M. Следуя ему, модель дополняется бинарной переменной f и ограничениями:

$$\begin{cases} x \ge f, \\ x < M \cdot f + 1. \end{cases}$$

Доказательство корректности ограничения $f = f_{im}(x)$ методом разбора случаев приведено в таблице 3. Из нее видно, оно выполняется тогда и только тогда, когда выполняются оба ограничения.

Таблица 3

x	f	$f_1 = (x \ge f)$	$f_2 = (\mathrm{x} < \mathrm{M} \cdot f + 1)$	$f_1 \wedge f_2$
0	0	true	true	true
0	1	false	true	false
(0;1)	0	true	true	true
(0;1)	1	false	true	false
1	0	true	false	false
1	1	true	true	true
(1; ∞)	0	true	false	false
(1; ∞)	1	true	true	true

Часть ограничений описана при помощи знаков строгого неравенства. Для программной реализации необходимо задание ограничений на значения переменной при помощи только нестрогих неравенств. С целью нивелирования данного обстоятельства используется константа $1/M$ - условно малое число. Тогда ограничение модели вида $x < y$ будет описано как $x + 1/M \leq y$.

Пример исходных данных для решения оптимизационной задачи

В качестве примера приводится математическая модель, размеры исходных матриц которой соответствуют следующим величинам:

$$l = 1, n = 5, m = 4, k = 3.$$

В качестве входных значений используются следующие матрицы:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}, D = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$E = (1 \quad 0 \quad 0 \quad 0 \quad 0).$$

Представление в виде графа, построенного в соответствии с описанными выше исходными данными модели, изображено на рисунке 1.

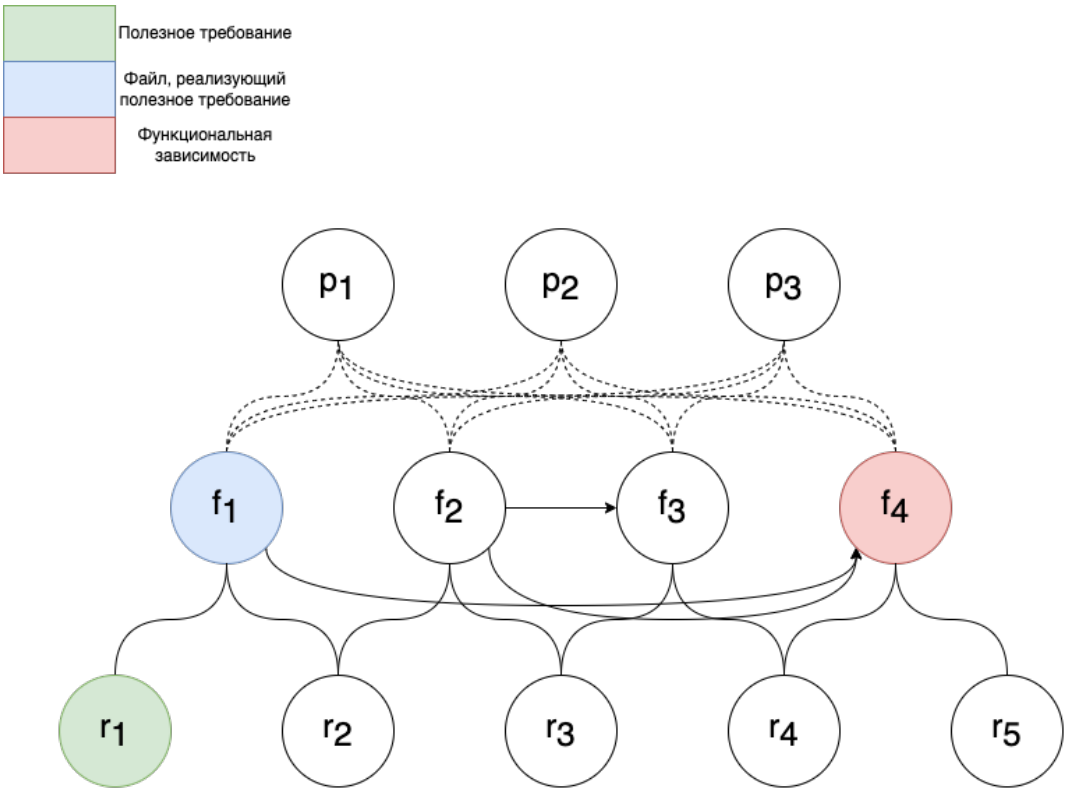

Рис. 1. Представление в виде графа

Построение математической модели по шагам:

1. Бинарные переменные матрицы A дополняют модель следующими ограничениями:

$$\begin{cases} a_{1,1} + a_{1,2} + a_{1,3} = 1, \\ a_{2,1} + a_{2,2} + a_{2,3} = 1, \\ a_{3,1} + a_{3,2} + a_{3,3} = 1, \\ a_{4,1} + a_{4,2} + a_{4,3} = 1. \end{cases}$$

2. Вектор полезных требований $R_{1\times5}$

$$R = (1 \quad 0 \quad 0 \quad 0 \quad 0).$$

3. Вектор полезных файлов исходного кода:

$$F = R \cdot T = (1 \quad 0 \quad 0 \quad 0 \quad 0) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} = (1 \quad 0 \quad 0 \quad 0).$$

4. Разрешение зависимостей полезных файлов исходного кода:

$$F \cdot \sum_{j=1}^{m=4} D^j = (0 \quad 0 \quad 1 \quad 0).$$

5. Полезные файлы исходного кода с разрешенными зависимостями:

$$\hat{F}_{1\times m} = F + F \cdot \sum_{j=1}^{m=4} D^j = (1 \quad 0 \quad 0 \quad 0) + (0 \quad 0 \quad 1 \quad 0) = (1 \quad 0 \quad 1 \quad 0).$$

6. Плагины, которые должны войти в поставку:

$$P = \hat{F} \cdot A = (1 \quad 0 \quad 1 \quad 0) \cdot \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \\ a_{4,1} & a_{4,2} & a_{4,3} \end{pmatrix} = (a_{1,1} + a_{3,1} \quad a_{1,2} + a_{3,2} \quad a_{1,3} + a_{3,3}).$$

Для расчета значений функции f_{in} выражения в матрице заменяются дополнительными бинарными переменными. $a_{1,1} + a_{3,1}$ заменяется на f_1 с добавлением в модели следующих ограничений:

$$\begin{cases} f_1 + 1/M - (a_{1,1} + a_{3,1} + 1) \le 0, \\ a_{1,1} + a_{3,1} - M \cdot f_1 \le 0. \end{cases}$$

$a_{1,2} + a_{3,2}$ заменяется на f_2 с добавлением в модели следующих ограничений:

$$\begin{cases} f_2 + 1/M - (a_{1,2} + a_{3,2} + 1) \le 0, \\ a_{1,2} + a_{3,2} - M \cdot f_2 \le 0. \end{cases}$$

$a_{1,3} + a_{3,3}$ заменяется на f_3 с добавлением в модели следующих ограничений:

$$\begin{cases} f_3 + 1/M - (a_{1,3} + a_{3,3} + 1) \le 0, \\ a_{1,3} + a_{3,3} - M \cdot f_3 \le 0. \end{cases}$$

Таким образом, плагины, которые должны войти в поставку, определяются вектором новых бинарных переменных:

$$P = (f_1, f_2, f_3).$$

7. Файлы исходного кода, которые должны войти в поставку:

$$\dot{F} = A \cdot P^T = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \\ a_{4,1} & a_{4,2} & a_{4,3} \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} a_{1,1} \cdot f_1 + a_{1,2} \cdot f_2 + a_{1,3} \cdot f_3 \\ a_{2,1} \cdot f_1 + a_{2,2} \cdot f_2 + a_{2,3} \cdot f_3 \\ a_{3,1} \cdot f_1 + a_{3,2} \cdot f_2 + a_{3,3} \cdot f_3 \\ a_{4,1} \cdot f_1 + a_{4,2} \cdot f_2 + a_{4,3} \cdot f_3 \end{pmatrix}.$$

Элементы матрицы $\dot{F}$ вычисляются нелинейными функциями, так как они содержат произведения переменных. Однако в силу того, что элементы матрицы A и переменные f_1, f_2 и f_3 бинарные, можно провести замену произведений дополнительными бинарными переменными.

$a_{1,1} \cdot f_1$ заменяется на f_4 с добавлением следующих ограничений:

$$\begin{cases} f_1 + a_{1,1} - (f_4 + 1) \leq 0, \\ f_4 - f_1 \leq 0, \\ f_4 - a_{1,1} \leq 0. \end{cases}$$

$a_{1,2} \cdot f_2$ заменяется на f_5 с добавлением следующих ограничений:

$$\begin{cases} f_2 + a_{1,2} - (f_5 + 1) \leq 0, \\ f_5 - f_2 \leq 0, \\ f_5 - a_{1,2} \leq 0. \end{cases}$$

$a_{1,3} \cdot f_3$ заменяется на f_6 с добавлением следующих ограничений:

$$\begin{cases} f_3 + a_{1,3} - (f_6 + 1) \leq 0, \\ f_6 - f_3 \leq 0, \\ f_6 - a_{1,3} \leq 0. \end{cases}$$

$a_{2,1} \cdot f_1$ заменяется на f_7 с добавлением следующих ограничений:

$$\begin{cases} f_1 + a_{2,1} - (f_7 + 1) \leq 0, \\ f_7 - f_1 \leq 0, \\ f_7 - a_{2,1} \leq 0. \end{cases}$$

$a_{2,2} \cdot f_2$ заменяется на f_8 с добавлением следующих ограничений:

$$\begin{cases} f_2 + a_{2,2} - (f_8 + 1) \leq 0 \\ f_8 - f_2 \leq 0 \\ f_8 - a_{2,2} \leq 0 \end{cases}$$

$a_{2,3} \cdot f_3$ заменяется на f_9 с добавлением следующих ограничений:

$$\begin{cases} f_3 + a_{2,3} - (f_9 + 1) \leq 0, \\ f_9 - f_3 \leq 0, \\ f_9 - a_{2,3} \leq 0. \end{cases}$$

$a_{3,1} \cdot f_1$ заменяется на f_{10} с добавлением следующих ограничений:

$$\begin{cases} f_1 + a_{3,1} - (f_{10} + 1) \le 0, \\ f_{10} - f_1 \le 0, \\ f_{10} - a_{3,1} \le 0. \end{cases}$$

$a_{3,2} \cdot f_2$ заменяется на f_{11} с добавлением следующих ограничений:

$$\begin{cases} f_2 + a_{3,2} - (f_{11} + 1) \le 0, \\ f_{11} - f_2 \le 0, \\ f_{11} - a_{3,2} \le 0. \end{cases}$$

$a_{3,3} \cdot f_3$ заменяется на f_{12} с добавлением следующих ограничений:

$$\begin{cases} f_3 + a_{3,3} - (f_{12} + 1) \le 0, \\ f_{12} - f_3 \le 0, \\ f_{12} - a_{3,3} \le 0. \end{cases}$$

$a_{4,1} \cdot f_1$ заменяется на f_{13} с добавлением следующих ограничений:

$$\begin{cases} f_1 + a_{4,1} - (f_{13} + 1) \le 0, \\ f_{13} - f_1 \le 0, \\ f_{13} - a_{4,1} \le 0. \end{cases}$$

$a_{4,2} \cdot f_2$ заменяется на f_{14} с добавлением следующих ограничений:

$$\begin{cases} f_2 + a_{4,2} - (f_{14} + 1) \le 0, \\ f_{14} - f_2 \le 0, \\ f_{14} - a_{4,2} \le 0. \end{cases}$$

$a_{4,3} \cdot f_3$ заменяется на f_{15} с добавлением следующих ограничений:

$$\begin{cases} f_3 + a_{4,3} - (f_{15} + 1) \le 0, \\ f_{15} - f_3 \le 0, \\ f_{15} - a_{4,3} \le 0. \end{cases}$$

В результате вектор $\dot{F}$ примет вид:

$$\dot{F} = \begin{pmatrix} f_4 + f_5 + f_6 \\ f_7 + f_8 + f_9 \\ f_{10} + f_{11} + f_{12} \\ f_{13} + f_{14} + f_{15} \end{pmatrix}.$$

8. Реализованные в поставке требования определяются по формуле:

$$\dot{R} = T \cdot \dot{F} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f_4 + f_5 + f_6 \\ f_7 + f_8 + f_9 \\ f_{10} + f_{11} + f_{12} \\ f_{13} + f_{14} + f_{15} \end{pmatrix} = \begin{pmatrix} f_4 + f_5 + f_6 \\ 0.5 \cdot (f_4 + f_5 + f_6) + 0.5 \cdot (f_7 + f_8 + f_9) \\ 0.5 \cdot (f_7 + f_8 + f_9) + 0.5 \cdot (f_{10} + f_{11} + f_{12}) \\ 0.5 \cdot (f_{10} + f_{11} + f_{12}) + 0.5 \cdot (f_{13} + f_{14} + f_{15}) \\ f_{13} + f_{14} + f_{15} \end{pmatrix}.$$

Для применения функция f_{im} к элементам $T \cdot \dot{F}$ вводятся дополнительные бинарные переменные. $f_4 + f_5 + f_6$ заменяется на f_{16} с добавлением следующих ограничений:

$$\begin{cases} f_{16} - (f_4 + f_5 + f_6) \le 0, \\ f_4 + f_5 + f_6 + 1/M - (1 + M \cdot f_{16}) \le 0. \end{cases}$$

$0.5 \cdot (f_4 + f_5 + f_6) + 0.5 \cdot (f_7 + f_8 + f_9)$ заменяется на f_{17} с добавлением следующих ограничений:

$$\begin{cases} f_{17} - (0.5 \cdot (f_4 + f_5 + f_6) + 0.5 \cdot (f_7 + f_8 + f_9)) \le 0, \\ 0.5 \cdot (f_4 + f_5 + f_6) + 0.5 \cdot (f_7 + f_8 + f_9) + 1/M - (1 + M \cdot f_{17}) \le 0. \end{cases}$$

$0.5 \cdot (f_7 + f_8 + f_9) + 0.5 \cdot (f_{10} + f_{11} + f_{12})$ заменяется на f_{18} с добавлением следующих ограничений:

$$\begin{cases} f_{18} - (0.5 \cdot (f_7 + f_8 + f_9) + 0.5 \cdot (f_{10} + f_{11} + f_{12})) \le 0, \\ 0.5 \cdot (f_7 + f_8 + f_9) + 0.5 \cdot (f_{10} + f_{11} + f_{12}) + 1/M - (1 + M \cdot f_{18}) \le 0. \end{cases}$$

$0.5 \cdot (f_{10} + f_{11} + f_{12}) + 0.5 \cdot (f_{13} + f_{14} + f_{15})$ заменяется на f_{19} с добавлением следующих ограничений:

$$\begin{cases} f_{19} - (0.5 \cdot (f_{10} + f_{11} + f_{12}) + 0.5 \cdot (f_{13} + f_{14} + f_{15})) \le 0, \\ 0.5 \cdot (f_{10} + f_{11} + f_{12}) + 0.5 \cdot (f_{13} + f_{14} + f_{15}) + 1/M - (1 + M \cdot f_{19}) \le 0. \end{cases}$$

$f_{13} + f_{14} + f_{15}$ заменяется на f_{20} с добавлением в модели следующих ограничений:

$$\begin{cases} f_{20} - (f_{13} + f_{14} + f_{15}) \le 0, \\ f_{13} + f_{14} + f_{15} + 1/M - (1 + M \cdot f_{20}) \le 0. \end{cases}$$

В результате вектор $\dot{R}$ примет вид

$$\dot{R} = \begin{pmatrix} f_{16} \\ f_{17} \\ f_{18} \\ f_{19} \\ f_{20} \end{pmatrix}.$$

9. Стоимость ППО комплектации:

$$f_c(E_1, A, T, D) = \dot{R}^T \cdot C \cdot \dot{R} = \begin{pmatrix} f_{16} \\ f_{17} \\ f_{18} \\ f_{19} \\ f_{20} \end{pmatrix}^T \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} f_{16} \\ f_{17} \\ f_{18} \\ f_{19} \\ f_{20} \end{pmatrix} =$$

$$= f_{16} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) + f_{17} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) + f_{18} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) + f_{19} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) + f_{20} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}).$$

Получившееся выражение нелинейно. Однако в силу того, что переменные f_{16}, f_{17}, f_{18}, f_{19} и f_{20} бинарные, можно привести выражение к линейному виду посредством использования дополнительных бинарных переменных.

$f_{16} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20})$ заменяется на f_{21} с добавлением следующих ограничений:

$$\begin{cases} f_{16} + f_{17} + f_{18} + f_{19} + f_{20} + f_{16} - (f_{21} + 1) \le 0, \\ f_{21} - f_{16} \le 0, \\ f_{21} - (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) \le 0. \end{cases}$$

$f_{17} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20})$ заменяется на f_{22} с добавлением следующих ограничений:

$$\begin{cases} f_{16} + f_{17} + f_{18} + f_{19} + f_{20} + f_{17} - (f_{22} + 1) \leq 0, \\ f_{22} - f_{17} \leq 0, \\ f_{22} - (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) \leq 0. \end{cases}$$

$f_{18} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20})$ заменяется на f_{23} с добавлением следующих ограничений:

$$\begin{cases} f_{16} + f_{17} + f_{18} + f_{19} + f_{20} + f_{18} - (f_{23} + 1) \leq 0, \\ f_{23} - f_{18} \leq 0, \\ f_{23} - (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) \leq 0. \end{cases}$$

$f_{19} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20})$ заменяется на f_{24} с добавлением следующих ограничений:

$$\begin{cases} f_{16} + f_{17} + f_{18} + f_{19} + f_{20} + f_{19} - (f_{24} + 1) \leq 0, \\ f_{24} - f_{19} \leq 0, \\ f_{24} - (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) \leq 0. \end{cases}$$

$f_{20} \cdot (f_{16} + f_{17} + f_{18} + f_{19} + f_{20})$ заменяется на f_{25} с добавлением следующих ограничений:

$$\begin{cases} f_{16} + f_{17} + f_{18} + f_{19} + f_{20} + f_{20} - (f_{25} + 1) \leq 0, \\ f_{25} - f_{20} \leq 0, \\ f_{25} - (f_{16} + f_{17} + f_{18} + f_{19} + f_{20}) \leq 0. \end{cases}$$

Результирующая целевая функция примет вид:

$$\min (f_{21} + f_{22} + f_{23} + f_{24} + f_{25}).$$

Полученная математическая модель включает 71 ограничение и 37 оптимизационных переменных. Из них переменных 12 являются переменными исходной задачи, а 25 – дополнительными переменными, введенными для приведения задачи к линейному виду.

Программная реализация

Описанная модель реализована на языке программирования Python. Реализация сочетает достоинства применения как парадигмы ООП, так и функционального программирования за счет активного использования лямбда-выражений.

Для реализации на языке программирования был использован внешний модуль Pyomo [10]. В части решаемой задачи этот модуль был задействован для построения математической модели и взаимодействия с решателями [10, 11]. Построение математической модели заключалось в динамической генерации переменных модели и ограничений. Взаимодействие с решателями - вызов их из переменных окружения среды, задание входных аргументов для их работы и считывание результатов вычислений.

С целью понижения сложности применяемого решения и достижения управляемости поведением каждой отдельной единицы функционала весь код

разбит на обособленные модули, представленные в виде классов языка программирования Python.

Компоновка их друг с другом, последовательное разрешение зависимостей происходит в едином месте - в главной функции программы. В такой схеме отсутствуют передача локальных неуправляемых переменных по цепочке вызовов в иерархии функций, а каждая отдельная единица функционала работает в своем информационном окружении. Диаграмма классов приведена на рисунке 2.

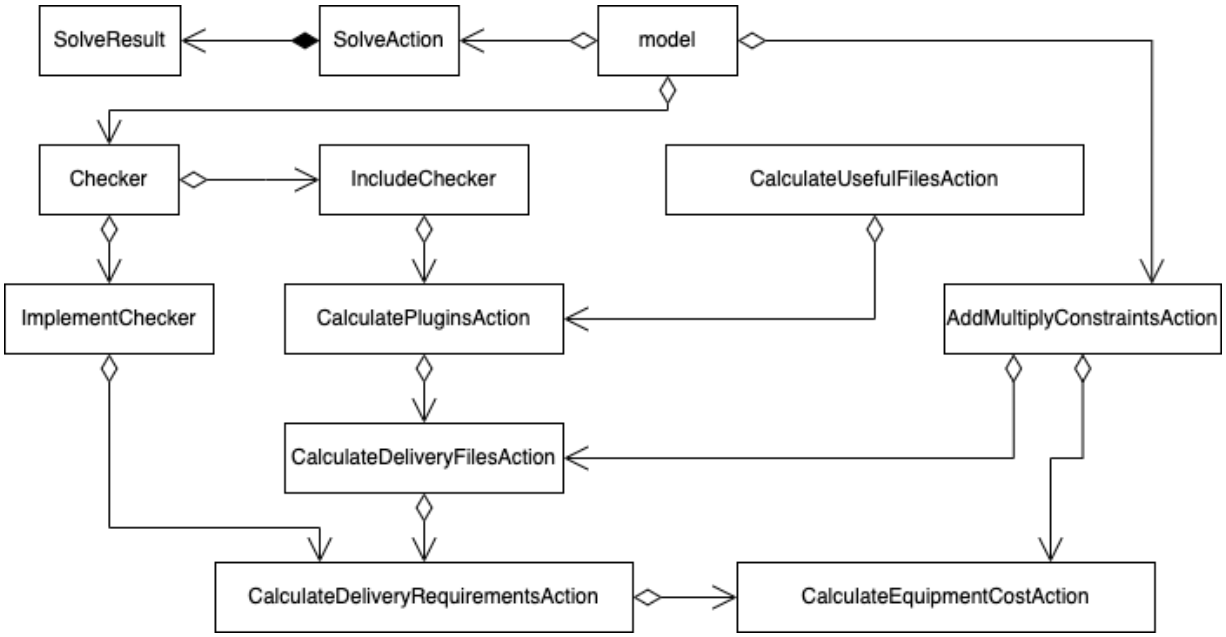

Рис. 2. Диаграмма классов

Вычислительные эксперименты

В целях верификации сформированной модели и оценки эффективности применения различных решателей проводился ряд экспериментов. В качестве входных данных для решателей использовалась сформированная ранее модель. Число переменных и ограничений на их значения в рассматриваемой модели позволяет задействовать решатели по открытой лицензии.

Характеристики оборудования, используемого для проведения экспериментов:

- операционная система - Ubuntu 23.04;
- процессор - 2-ядерный процессор Intel Core i5 с тактовой частотой 1,8GHz;
- объем ОЗУ - 8 ГБ.

Были использованы следующие решатели: glpk, gurobi, xpress, gams, gdpopt, mindtpy, cbc, conopt, copt, cplex, ilogcp и minos.

В рамках проведения экспериментов каждый решатель находил решение задачи 100 раз. При каждом расчете вычислялось результирующее значение

стоимости ППО (см. табл. 4), а также замерялось время, необходимое решателю для выполнения вычислений (см. рис. 3).

Таблица 4

Решатель	Стоимость ППО
glpk	1
gurobi	1
xpress	1
gams	1
gdpopt	1
mindtpy	1
cbc	1
conopt	25
copt	1
cplex	1
ilogcp	1
minos	25

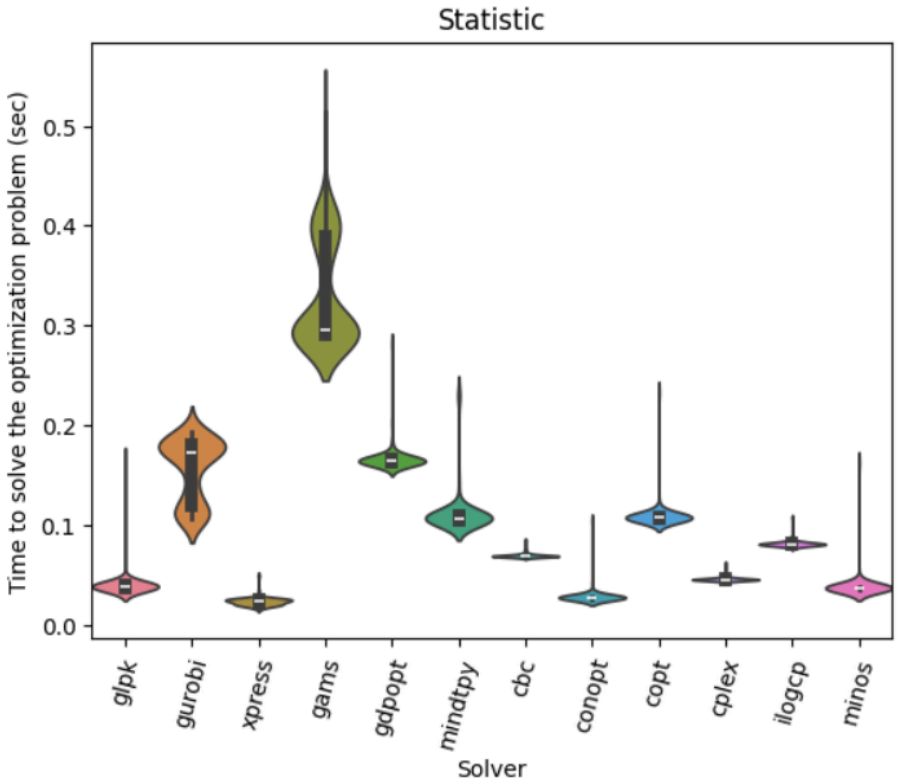

Рис. 3. Данные о потребном времени работы решателей

Результаты экспериментов показывают, что время, необходимое различным решателям для вычисления одних и тех же параметров, может

отличаться в разы. Кроме того, некоторые программы не смогли найти оптимальное решение. Наименьшее значение стоимости ППО, полученное в ходе экспериментов, составляет 1. Это значение было достигнуто при применении результатов работы 10 решателей из 12.

Заключение

В статье предложена математическая модель для решения задачи оптимальной декомпозиции программного обеспечения на плагины. Используя ее, можно получать оптимальное распределение файлов исходного кода по плагинам с целью минимизации стоимости ППО реализованного функционала в поставке заданной комплектации. Данные сведения могут быть полезны как при планировании архитектуры приложения, так и при формировании заявок комплектаций готовых версий программного обеспечения.

В дальнейших исследованиях предполагается изучение зависимости времени работы решателей от размера решаемой задачи: количества параметров и ограничений в ней.

Библиографический список

1. Василенко В.В., Рыженко С.В. Организация безопасного файлового обмена между корпоративной сетью и сетью общего пользования // Ежеквартальный рецензируемый, реферируемый научный журнал «Вестник АГУ». Вып. 2 (321). 2023. С. 40-46.

2. Архипов А.Е., Карпушкин С.В. Структурно-параметрический синтез систем визуализации для тренажерных комплексов // Вестник Тамбовского государственного технического университета 2022. Том 28. № 3. С. 428-443.

3. Демушкина К.М., Кузьмин А.В. Анализ возможностей инструментов реализации технологии process mining // Известия Самарского научного центра Российской академии наук. Том 25. № 4. 2023. С. 114-120.

4. Тюменцев Е.А., Згуровец Д.Ю. Алгоритм загрузки плагинов, не имеющих явных зависимостей между собой // Математические структуры и моделирование. 2021. № 1(57). С. 101–107.

5. Костромин Р.О. Сравнительный обзор средств управления конфигурациями ресурсов вычислительной среды функционирования цифровых двойников // «Information and mathematical technologies in science and management». 2021. № 1 (21). С. 132-145.

6. Добрянский Г.В., Мельникова Н.С., Мовила В.Н. Инструментальная программная платформа для разработки информационно-диагностических комплексов ИПП «Салют» // Вестник УГАТУ. 2022. Том 26. № 3 (97). С. 90-99.

7. Лаврищева Е.М., Зеленов С.В. Модельный подход к обеспечению безопасности и надежности Web-сервисов // Труды ИСП РАН. Том 32. Вып. 5. 2020. С. 153-166.

8. Горбунов-Посадов М.М. Как растет программа // Препринты ИПМ им. М.В.Келдыша. 2000. № 50. 16 с.

9. Богданов И.П., Нестеров В.А., Судаков В.А., Сыпало К.И., Топоров Н.Б. Расчет оптимальной загрузки воздушных транспортных средств с учетом приоритизации летательных аппаратов // Известия РАН. Теория и системы управления. 2021. № 3. С. 57–70.

10. Сивакова Т.В., Судаков В.А., Шимко В.С. Исследование методов решения задач смешанного целочисленного линейного программирования // Препринты ИПМ им. М.В.Келдыша. 2024. № 24. 18 с.

11. Белозеров И.А., Судаков В.А. Машинное обучение с подкреплением для решения задач математического программирования // Препринты ИПМ им. М.В.Келдыша. 2022. № 36. 14 с.

Оглавление

Введение 3
Описание модели 3
Пример исходных данных для решения оптимизационной задачи 8
Программная реализация 13
Вычислительные эксперименты 14
Заключение 16
Библиографический список 16