ISSN 2658-5014 (Print)
2022. Т. 4, № 2 (9). С. 43–48
СИИТ
СИСТЕМНАЯ ИНЖЕНЕРИЯ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
ISSN 2686-7044 (Online)
http://siit.ugatu.su

УДК 519.85
doi 10.54708/26585014_2022_42943

Возможность частичной реализации при алгоритмическом подходе и большом списке сущностей

А. Р. Ураков [1], Г. И. Федорова [2]

[1] urakov05@gmail.com, [2] g_fed@mail.ru

ФГБОУ ВО «Уфимский государственный авиационный технический университет» (УГАТУ)

Поступила в редакцию 14 сентября 2022 г.

Аннотация. Перед началом работ над крупным программным проектом оценивается трудоемкость будущей реализации. Такая оценка имеет основной целью привлечение средств, поэтому всегда делается настолько оптимистичной, насколько в нее вообще можно поверить. Предполагается, что в ходе работы над проектом, по мере накопления опыта и появления каких-то промежуточных результатов, будет сделана новая более реалистичная оценка и планы будут соответствующим образом скорректированы. Такая оценка становится особенно важной, если завершение каких-то этапов отстает от графика, так как появляется подозрение, что отставание может быть вызвано не организационными, а фундаментальными математическими причинами, поэтому оно не может быть преодолено. В работе предлагается простой способ не только сделать такую оценку для разрабатываемых проектов, но и оценить вероятность реализации сложных проектов в целом. В первом случае предлагается использовать такой экспериментальный критерий как отношение количества ошибочных решений на общее количество решений, которое сделала система управления при работе в требуемых условиях. Такой критерий обычно легко доступен разработчикам в ходе разработки и отладки проекта. Во втором случае предлагается исходить из приближенной оценки общего необходимого количества сущностей.

Ключевые слова: алгоритмический подход; сложные задачи; сущность; частичная реализация; полная реализация; ошибка принятия решения; распределение Парето; возможность реализации; трудоемкость разработки; оценка сложности разработки преобразования отношений.

ВВЕДЕНИЕ

В настоящее время существует два принципиально разных подхода к реализации сложной управляющей системы: классический (или алгоритмический) и нейросетевой [1, 2].

Оба подхода предполагают наличие некоторого множества программ (подпрограмм) таких, что каждая:

а) принимает на вход данные с датчиков или результаты выполнения других программ;

б) в результате выполнения формирует управляющее решение или передает результат в качестве входа другим программам. Разница между подходами в том, как создаются эти программы.

Здесь пойдет речь о первом подходе. Он подразумевает, что каждая подпрограмма рассматривает некое предварительно формализованное состояние одного из объектов системы (или набор таких состояний). Для каждого такого состояния сформированы

и представлены в виде алгоритма правила принятия решения. Подпрограмма выполняет заданный алгоритм, выдавая одно из возможных решений. Каждый такой объект назовем сущностью. Сущностью также будем называть каждое из возможных состояний и каждое используемое правило.

Классический подход предполагает, что все данные и алгоритмы в системе должны быть представлены в виде сущностей. Тогда алгоритмы (реализованные в виде машинного кода) получают некоторые сущности на вход, передают сущности на выход и сами при этом являются сущностями в управляющей системе.

Отметим еще одну особенность реализации при классическом подходе. Если подпрограмма выдает решение, которое нас не устраивает, нужно либо скорректировать текущую сущность, либо добавить новую. Возможно, появление новой сущности приведет к добавлению новой подпрограммы.

Задачу управления можно сформировать как задачу преобразования множества возможных входов X в множество сигналов управления Y или $X \rightarrow Y$. Тогда полное решение задачи – это преобразование каждого элемента множества X в корректное Y. Для полного решения требуется реализация некоторого множества сущностей E. Здесь и далее под реализацией сущностей будем понимать их строгое (формальное) описание, представление в виде алгоритмов или данных, кодирование.

Несомненно, существуют системы, для которых задача может быть реализована полностью, но нам интересны такие случаи, когда подобное невозможно. Например, это задача управляющей системы беспилотного автомобиля. Во-первых, не все ситуации строго формализованы, во-вторых, ранее было показано, что сложность точного решения задачи не ниже, чем PSPASE [3–5]. Из практики известно, что количество сущностей, которые требуется реализовать для полноценного применения, крайне велико. Только в ПДД [6] можно насчитать тысячи терминов. При этом большая часть используемых терминов (например, «твердое покрытие», «примыкание», «препятствие», «угроза») в ПДД или не определена или определена довольно грубо. В результате, каждый из таких терминов при попытке его строгой формализации выражается очень большим количеством (десятки, сотни и тысячи) новых сущностей. Так же нужно учесть, что необходимость одновременного применения разных правил превращает в сущность их комбинацию. Дополнительный источник сущностей в том, что большое количество правил, необходимых для принятия решения, не описаны в ПДД, а предполагаются как очевидные для водителя.

Возможен другой способ оценки количества сущностей – как общее количество функций, переменных и множеств значений переменных, влияющих на выбор решения [7]. Практика показывает, что при реализации практических задач управления это количество может исчисляться миллионами [8].

ЗАДАЧА ЧАСТИЧНОЙ РЕАЛИЗАЦИИ

Для всех задач достаточно высокой сложности ставится задача частичной реализации. Частичная реализация означает, что система в состоянии правильно обработать некоторую часть входной информации X_r (т.е. для любого $x \in X_r$ выдать корректное управляющее решение $y \in Y$). Введем отношение $s_r = |X_r|/|X|$, здесь s_r – это уровень реализации.

Можно поставить задачу частичной реализации таким образом, чтобы уровень реализации был не ниже, чем некий пороговый s_r. Однако, в практическом смысле важнее такая ситуация, при которой ошибочные решения встречаются не чаще, чем некоторое пороговое $\varepsilon_r = 1 - s_r$.

Например, для задачи управления беспилотным автомобилем по самым простым оценкам [9] предполагается допустимый уровень ошибочных решений не чаще, чем одна ошибка на 10^8 миль пробега. Так же приблизительно предполагая необходимость не менее одного управляющего реше-

ния каждую секунду или каждые несколько метров, мы получаем допустимый уровень ошибок ε_r порядка 10^{-10}.

Попытаемся оценить минимально возможную сложность частичной реализации такого высокого уровня. Для этого далее будем предполагать, что реализация происходит оптимальным образом, т.е. таким, при котором все отклонения от заданного порядка будут увеличивать затраты ресурсов и времени на реализацию.

Для частичной реализации требуется реализовать не все, а только некоторую часть сущностей E_r. Реализация определенного объема сущностей позволяет получить корректное решение для определенного объема входных данных. Другими словами, если мы реализовали группу сущностей E_i, это означает, что множество входов, для которых управляющее решение принимается корректно, увеличивается на подмножество X_i.

Реализация происходит поэтапно: реализуется подмножество E_1, затем E_2, E_3, … и т.д. На каждом этапе увеличивается мощность X_r. Сначала реализуется E_1, которая позволяет получить верные решения для подмножества X_1 мощностью q_1, затем E_2, что совместно с E_1 дает $X_r = X_1 \cup X_2$ мощностью $s_2 = q_1 + q_2$. На следующем этапе мощность X_r достигает $s_3 = q_1 + q_2 + q_3$, … и т.д. Предполагается, что при идеально организованном процессе уровень реализации только увеличивается.

Из практики известно, что реализация разных сущностей приводит к реализации разных объемов входа, т.е. предположение, что $q_i = q_j$ выполняется для любых i, j – неверно. Это можно обосновать, но, не теряя времени, будем считать это аксиомой. Заметим, что изменение последовательности реализаций может приводить к изменению значений некоторых из q_i, однако, в нашем идеальном случае это изменение не уменьшает уровня реализации.

Из предположения неравенства q_i следует, что все q могут быть отсортированы от большего к меньшему. При идеальном подходе сущности реализуются в таком порядке, что если $q_i > q_j$, то подмножество сущностей E_i реализуется раньше, чем E_j. Несложно показать, что изменение приведенного порядка не может увеличить уровень реализации для некоторого реализованного списка.

Таким образом, мы распределяем номера подмножеств E_i таким образом, что если $i < j$, то $q_i > q_j$. Мы получили некоторый ряд вида $q_i = q(i)$. Ряд положительный, строго убывающий, а частная сумма элементов ряда равна единице, т.е.:

$$\sum_{i=1}^{n} q_i = 1. \qquad (1)$$

На практике, для подобных случаев, широко применяется распределение Парето (Бредфорда) [10], которое в нашем случае выглядит так:

$$q_i = \frac{a}{i^{k+1}}, \qquad (2)$$

где $k > 0$ и функция вероятности получает вид:

$$s_i = 1 - \frac{a}{ki^k}. \qquad (3)$$

В таком распределении значения q обратно пропорциональны номеру в последовательности, значит производная ds_i/di не просто обратно пропорциональна номеру элемента, но и крайне мала для больших i, т.е. s_i близких к 1 или ε_i близких к нулю.

СЛОЖНЫЕ ЗАДАЧИ И СПОСОБЫ ИХ РЕШЕНИЯ

Такая зависимость приводит к следующему эффекту. Например, при $k = 1$ для того, чтобы уменьшить ошибку ε_r в два раза, необходимо реализовать такое же количество сущностей, которое уже было реализовано до этого. Это же означает, если в какой-то момент реализация s_r достигла 90 %, то достижение требуемых 10^{-10} приводит к необходимости реализовать в 10^9 раз более сложную задачу, чем уже реализованная, что, конечно, практически немыслимо.

Замена k на другие величины, при сохранении распределения Парето, принципи-

ально ситуацию не меняет и реализацию более реальной не делает.

Какие возможны выходы из такой ситуации. Рассмотрим их.

– Ограниченное количество сущностей. Если список сущностей достаточно мал, можно реализовать их все и таким образом решить задачу. Собственно, так и происходит в стандартных задачах управления простой системой.

– Более грубая реализация или изменение условий реализации. Некоторые реализации предполагают кардинально уменьшить требования к уровню ошибки, задавая ее такой, какая уже достигнута. Это можно сделать таким образом: если решение приводит к ошибке, к управлению подключается человек. Другой подход заключается в том, чтобы ограничивать условия применения управляемой системой до тех пор, пока реализованный набор сущностей не будет в состоянии справиться с системой, соблюдая заданный уровень ошибки. Недостаток здесь в том, что ограничений может понадобиться так много, что устройство, полученное для таких условий, окажется бесполезным.

– Использование другого распределения весов сущностей. Простая замена одного распределения на другое, например, замена распределения Парето на распределение такого вида $q_i = a/k^i$ позволяет быстро достичь требуемого уровня ε_r однако делает это малым числом сущностей, поэтому означает тоже, что и 1) и не подходит по условиям задачи.

– Можно предположить наличие некоторого особого распределения, устроенного следующим образом. Оно имеет распределение вида $q_i = a/k^i$ до достижения функцией s требуемого уровня реализации s_r и вида $q_i = a/i^{k+1}$ после нее. Такое распределение позволяет достичь требуемого уровня ошибки на малом числе сущностей, но весь остальной огромный массив сущностей попадает в область вне ошибки. Другими словами, такое распределение предполагает, что при надежной реализации встретится только малая часть всех сущностей, а остальные необязательны. Например, основные правила ПДД нужно реализовать, а остальные все равно не используются. Такой подход выглядит странным как из практических соображений, так и по самой формулировке распределения, которое выглядит достаточно надуманным. В частности, из всех возможных вариантов распределения чудесным образом выбирается именно такой, который требуется для реализации задачи.

– Сложность реализации новых сущностей экспоненциально уменьшается по мере перехода от более часто встречающихся к более редким. Рассчитывать на такое крайне сложно, так как обычно происходит обратное – часто встречающиеся сущности проще редких. Здесь мы для простоты полагаем, что сложность реализации примерно одинакова. При этом из [11] следует, что сложность реализации по мере работы над системой управления будет возрастать.

ОЦЕНКА ВОЗМОЖНОСТИ РЕАЛИЗОВАТЬ СЛОЖНУЮ ЗАДАЧУ

Из изложенного во введении следует, что для определенного круга задач частичная реализация будет единственно возможным решением. В частности, для задачи управления беспилотным транспортом.

Рассмотрим случай, при котором требуется алгоритмически реализовать задачу высокой сложности. Как оценить возможность реализовать задачу, если уже накоплен какой-то опыт? Как она будет реализована, если такое возможно? Мы предлагаем следующие шаги, позволяющие сделать такую оценку.

– Необходимо оценить проблему, на наличие в ее составе задач сложности NP-hard [12]. Если таковые присутствуют, необходимо или исключить их полностью, или заменить эвристиками, которые смогут обеспечить решение достаточной точности. Если такое исключение произошло, то проблема переходит в другую предметную область (это не гарантирует, что задача может быть решена, но исключает из нашей классификации), если хотя бы одна подзадача осталась, переходим к следующему пункту.

– Для каждой из оставшихся NP-hard задач и всех узлов, которые используют ре-

зультаты решения этих задач, требуется предусмотреть корректный выход на случай ошибки (перегрузка, передача управления человеку и т.д.). Возможен один выход для многих задач. Если такой выход допустим, реализация делается с опорой на этот выход. Если такое невозможно, переходим к следующему пункту.

– Определяется допустимая ошибка управления, т.е. такая, которая позволит эксплуатировать разработку. Выполняется частичная реализация, которая укладывается в данную ошибку. Частичная реализация предполагает ограничения на условия применения разработки. Эксплуатация при таких ограничениях имеет смысл? Если да, выпускается устройство для данных условий. Если нет, переходим к следующему пункту.

– Мы имеем допустимую ошибку управления ε_r, ошибку готовой частичной реализации ε_t и количество сущностей, реализованных в ней $|E_t|$. Тогда количество сущностей $|E_r|$, которое нужно реализовать для достижения уровня ε_r, настолько выше $|E_t|$, насколько ε_t больше ε_r, т.е.:

$$|E_r| \approx |E_t| \frac{\varepsilon_t}{\varepsilon_r}. \quad (4)$$

Допуская, что в идеальном случае затраты (в человеко-часах) на реализацию каждой сущности примерно одинаковы, получаем что необходимый уровень затрат C_r связан с уже произведенными затратами C_t формулой

$$C_r \approx C_t \frac{\varepsilon_t}{\varepsilon_r}. \quad (5)$$

На практике трудоемкость искомой реализации будет выше, чем наша оценка требуемого числа сущностей, так как приходится учитывать необходимость согласовывать сущности с уже существующими. В частности, происходит экспоненциальный рост трудоемкости относительно увеличения числа сущностей [11].

Практическое применение оценки выглядит следующим образом. Возьмем случай, когда внедрение устройства требует уровня ошибок не выше 10^{-7}. Затратив определенные ресурсы (финансы, время), для заданных условий применения удалось достичь уровня 10^{-3}. Это означает, что попытка достичь требуемого уровня потребует вложений, как минимум, в 10 тысяч раз больших, чем те, что уже сделаны.

ЗАКЛЮЧЕНИЕ

Приведенная оценка рассматривает оптимистичный случай, при котором процесс реализации значительно идеализирован. В частности, предполагается, что программирование, а именно превращение алгоритмов в программный код происходит без ошибок, а его внедрение в систему происходит без влияния на работоспособность других подпрограмм. При реальной разработке такое маловероятно, поэтому оценка трудоемкости несмотря на то, что выглядит крайне высокой (как на последнем примере), может оказаться заниженной. Тем не менее, даже в таком виде, она позволяет получить более реалистичный взгляд на перспективы завершения многих сложных проектов.

СПИСОК ЛИТЕРАТУРЫ

1. **Нейронная** сеть. Большая российская энциклопедия: [в 35 т.] / гл. ред. Ю. С. Осипов. М.: Большая российская энциклопедия, 2004–2017. **[** Yu. S. Osipov (ch. ed.), Neural network. Great Russian Encyclopedia: [in 35 volumes], (in Russian). Moscow: Great Russian Encyclopedia, 2004–2017. **]**

2. **Машины** Тьюринга и рекурсивные функции / Г.-Д. Эббинхауз [и др.]; пер. с нем. Э. Г. Белаги. М.: Мир, 1972. 264 с. **[** G.-D. Ebbinhouse, *et al.*; translated from German by E. G. Belagi, *Turing machines and recursive functions,* (in Russian). Moscow: Mir, 1972. **]**

3. **Reif J., Sharir M.** Motion Planning in the Presence of Moving Obstacles // 26th Annual Symposium on Foundations of Computer Science. Portland: IEEE, 1985. Pp. 144-154.

4. **Canny J., Reif J**. New Lower Bound Techniques for Robot Motion Planning Problems // 28th Annual Symposium on Foundations of Computer Science. Los Angeles: IEEE, 1987. Pp. 49-60.

5. A Review of Motion Planning Tech-niques for Automated Vehicles / D. González, et al. // IEEE Transactions on Intelligent Transportation Systems. 2016. Vol. 17. Pp. 1135-1145.

6. **Постановление** Правительства РФ от 23.10.1993 № 1090 (ред. от 31.12.2020) «О Правилах дорожного движения» (вместе с «Основными положениями по допуску транспортных средств к эксплуатации и обязанности должностных лиц по обеспечению безопасности дорожного движения») (с изм. и доп., вступ. в силу с 01.01.2022) // Постановление Правительства РФ. 31.12.2020. № 2441.

7. **Холстед М. Х.** Элементы науки о программном обеспечении. Амстердам: Elsevier North-Holland, 1977. 128 p. **[** M. H. Halsted, *Elements of software science*, (in Russian). Amsterdam: Elsevier North-Holland, 1977. **]**

8. **Котов С. Л., Палюх Б. В., Федченко С. Л.** Разработка, стандартизация и сертификация программных средств и информационных технологий и систем: учеб. пособие для вузов по спец. "Прикл. информатика (по обл.)" и др. экон. спец. Тверь: ТвГТУ, 2006. 103 с. **[** S. L. Kotov, B. V. Palyukh, S. L. Fedchenko, *Development, standardization and certification of software tools and information technologies and systems: textbook. allowance for universities on special. "Applied Informatics (by region)" and other economics. Specialist,* (in Russian). Tver: TVGTU, 2006. **]**

9. **Kalra N., Paddock S.** Driving to Safety: How Many Miles of Driving Would it Take to Demonstrate Autonomous Vehicle Reliability? // Transportation Research Part A: Policy and Practice. 2016. Vol. 94. Pp. 182-193.

10. **Распределение** Парето. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Распределение Парето (дата обращения 18.07.2022). **[** Pareto distribution (2022, Jul. 18), [Online]. Available: https://ru.wikipedia.org/wiki/ Распределение Парето **]**

11. **Ураков А. Р., Тимеряев Т. В.** Актуальные проблемы автоматического управления транспортными средствами // Интеллектуальные технологии на транспорте. 2021. № 1 (25). С. 35–45. **[** A. R. Urakov, T. V. Timeryaev, “Actual problems of autonomous vehicle control”, (in Russian), in *Intellektualnye tehnologii na transporte*, no. 1 (25), pp. 35-45, 2021. **]**

12. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с. **[** M. Gary, D. Johnson, *Computing machines and hard-to-solve problems*, (in Russian). Moscow: Mir, 1982**.]**

ОБ АВТОРАХ

УРАКОВ Айрат Ренатович, доц. каф. вычислительной математики и кибернетики. Дипл. инженер-системотехник (МГТУ им. Баумана, 1993). Канд. физ.- мат. наук по примен. выч. техн., мат. мод. и мат. мет. в научн. исслед. (БГУ, 1997). Иссл. в обл. мат. мод. в электрохимическом формообразовании, теории графов, теории алгоритмов.

ФЕДОРОВА Галина Ильясовна, доц. каф. высокопроизводительных вычислительных технологий и систем. Дипл. инженер-математик (УГАТУ, 2000). Канд. физ.- мат. наук по мат. мод., числ. мет. и компл. программ (УГАТУ, 2004). Иссл. в обл. мат. мод. процессов электрохимического формообразования.

METADATA

Title: Partial implementation capability with algorithmic approach and large list of entities.

Authors: A. R. Urakov [1], G. I. Fedorova [2]

Affiliation: Ufa State Aviation Technical University (UGATU), Russia.

Email: [1] urakov05@gmail.com, [2] g_fed@mail.ru

Language: Russian.

Source: SIIT (scientific journal of Ufa State Aviation Technical University), vol. 4, no. 2 (9), pp. 43-48, 2022. ISSN 2686-7044 (Online), ISSN 2658-5014 (Print).

Abstract: Before starting work on a large software project, the complexity of future implementation is assessed. Such an assessment has the main goal of raising funds, so it is always made as optimistic as you can even believe in it. It is anticipated that during the work on the project, as experience is gained and some intermediate results emerge, a new, more realistic assessment will be made and the plans will be adjusted accordingly. Such an assessment becomes especially important if the completion of some stages is behind schedule, as there is a suspicion that the lag may be caused not by organizational, but by fundamental mathematical reasons, so it cannot be overcome. The paper offers a simple way not only to make such an assessment for the projects being developed, but also to assess the likelihood of implementing complex projects as a whole. In the first case, it is proposed to use such an experimental criterion as the ratio of the number of erroneous decisions to the total number of decisions that the control system made when working in the required conditions. Such a criterion is usually easily accessible to developers during the development and debugging of the project. In the second case, it is proposed to proceed from an approximate estimate of the total required number of entities.

Key words: algorithmic approach; complex tasks; entity; partial implementation; full implementation; decision-making error; Pareto distribution; possibility of implementation; labor intensity of development; Assess the complexity of the development transformation.

About authors:

URAKOV, Airat Renatovich, Assoc. Prof., Dept. of Computational Mathematics and Cybernetics, Computer Science and Robotics Faculty (USATU). Dipl. Systems technics engineer (MSTU, 1993). Cand. of Phis.-Math. Sci. (BSU, 1997).

FEDOROVA, Galina Ilyasovna, Assoc. Prof., Dept. of High-performance computing technologies and systems, Computer Science and Robotics Faculty, (USATU). Dipl. Mathematics engineer (USATU, 2000). Cand. of Phis.-Math. Sci. (USATU, 2004).