DOI 10.37882/2223–2966.2023.01.35

РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ ВЫСОКОЙ СЛОЖНОСТИ. ПОДХОДЫ И УСЛОВИЯ

IMPLEMENTATION OF A HIGH COMPLEXITY CONTROL SYSTEM. APPROACHES AND CONDITIONS

A. Urakov
S. Porechny

Summary. The features of the Turing machine are considered, which does not allow the development of a reliable control system with a sufficiently large number of inputs. Based on these features, a set of four conditions is proposed which a new non-Turing machine must fulfill in order to implement reliable control in an environment of any complexity.

The paper shows two types of development on a Turing machine: classic and neural networks. The first two of the formulated conditions are satisfied with the neural networks, but not with the classic approach. The third is performed only with the classic, but not with the neural networks approach. The fourth, in its formulation, is not satisfied with the neural networks approach, but additionally requires the fulfillment of the first two conditions, which means that it is not satisfied with the classic approach either.

Keywords: complex systems, Turing machine, non-turing machine, conditions for the implementation of a complex system, algorithmic control, neural network control.

Ураков Айрат Ренатович
К.ф.-м.н., доцент, ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ), г. Уфа
urakov@ufanet.ru
Поречный Сергей Сергеевич
К.ф.-м.н., доцент, ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ), г. Уфа
porechny@mail.ru

Аннотация. Рассматриваются особенности машины Тьюринга, которые не позволяют разработать надежную управляющую систему при достаточно большом числе входов. На основе этих особенностей предлагается набор четырех условий, которые должна выполнить новая нетьюринговая машина, для реализации надежного управления в среде любой сложности.

В работе показаны два типа разработки на машине Тьюринга: классический и нейросетевой. Первые два из сформулированных условий выполняются при нейросетевом, но не классическом подходе. Третье выполняется только при классическом, но не нейросетевом подходе. Четвертое по своей постановке не выполняется при нейросетевом, но дополнительно требует выполнения первых двух условий, а значит, не выполняется и при классическом подходе.

Ключевые слова: сложные системы, машина Тьюринга, нетьюринговая машина, условия реализации сложной системы, алгоритмическое управление, нейросетевое управление.

Введение

В настоящее время существует единственный способ решения сложных управляющих задач — на базе вычислительной машины (компьютера) и соответствующего программного обеспечения. Управляющая программа получает на вход информацию с датчиков и выдает сигналы органам управления. Такой способ предполагает наличие машинного кода и необходимых ему данных, размещенных в ячейках памяти. Код, в свою очередь, представляет собой преобразование определенных алгоритмов на язык инструкций, доступных вычислительной машине. [1, с. 7–32]

Введем необходимые обозначения. Пусть $\boldsymbol{X}$ — множество возможных входных сигналов, а x — текущее состояние датчиков, $x \in \boldsymbol{X}$. Множество управляющих сигналов (УС) обозначим как $\boldsymbol{Y}$, тогда y — текущий УС, $y \in \boldsymbol{Y}$. Управляющая машина выполняет некоторую функцию вида f:

$\boldsymbol{X} \rightarrow \boldsymbol{Y}$ или $y = f(x)$.

Такая функция может быть реализована в виде таблицы соответствий (x_i, y_j), т.е. такой таблицы, в которой для каждого варианта входных данных x_i указано правильное управляющее решение y_j. Такой подход успешно применяется на практике для достаточно простых систем. В сложных УС размер множества входов $\boldsymbol{X}$ будет комбинаторно зависеть от размерности входных данных и может оказаться экстремально большим.

Рассмотрим следующий пример. Автомобиль, управляемый автопилотом, оснащен камерами, передающими изображение. Уже сейчас для принятия решения используется несколько камер и ретроспектива (несколько снимков со сдвигом по времени для учета динамики). Пусть для этого требуется 2^5 кадров размером в 1 мегабайт или 2^{20} (для современных камер это число даже выше). Это означает, что входная инфор-

мация в общем случае имеет размер 225 бит. Отсюда, теоретическая мощность $\boldsymbol{X}$ может достигать $2^{2^{25}}$, т.е. примерно $2^{30000000}$. Разместить в памяти таблицу соответствий такого размера, как и дождаться завершения поиска по ней, практически невозможно.

Заметим, что в практических задачах очень большие группы входных данных имеют общее управляющее решение. Например, если управляющее устройство понимает всего 2 команды (например: вперед и назад), то это означает, что все множество входных данных $\boldsymbol{X}$ может быть разбито всего на 2 группы. Осталось только найти «границы» между ними. [2]

Если провести границы просто, то и задача управления оказывается простой вне зависимости от размера $\boldsymbol{X}$. Простота предполагает, например, что для принятия решения достаточно оценить небольшую часть (несколько бит) входного слова. Когда говорят о сложной системе, предполагают, что не существует простого способа разделить множество $\boldsymbol{X}$ по видам управления. Например, если для каждого варианта входа корректное решение было задано случайным образом, то поиск решения в любом случае сводится к просмотру всей таблицы, что, как указано выше, невозможно для больших входов. [2]

В практических задачах, к счастью, корректное решение не задается случайно. УС отражает реалии окружающего мира, который состоит из объектов и правил их взаимодействия. Следовательно, решение может быть вычислено гораздо быстрее с помощью этих правил. Эффективно используются следующие особенности реального мира.

Первая. Количество реальных объектов намного меньше, чем состояний, которые они могут генерировать. Более того, при решении конкретной задачи управления выясняется, что различные объекты можно рассматривать как одинаковые. Это позволяет еще сильнее уменьшить размер входа. Для этого нужно распознавать объекты, по возможности группируя их.

Вторая. Объекты всегда генерируют состояния по правилам. Эти правила могут быть представлены в виде формул (точных и приближенных), таблиц, ограничений и т.д. Так как текущее состояние формируется комбинаторным взаимодействием разных правил, общее количество правил комбинаторно меньше числа состояний, которые они формируют. Следовательно, нужно формировать управляющее решение с учетом этих правил.

Третья. В конкретной ситуации участвуют не все объекты, а только малая их часть. Значит, для принятия решения в каждом конкретном случае, нужно рассматривать только небольшую долю всех объектов.

Из этих особенностей следует, что систему управления возможно реализовать не в виде одной функции, а с помощью множества промежуточных функций $\boldsymbol{G}$ вида gi: $\boldsymbol{Pi\ Qi}$. Размерность аргумента каждой такой функции должна быть небольшой. В него могут входить как часть входных данных, так и какие-то из результатов выполнения других функций. Каждая функция не только выдает промежуточное решение, но и указывает, какие функции следует вызвать для обработки этих решений. Так происходит до тех пор, пока не будет получено управляющее решение.

В такой системе управления:

а) нет сложных функций, которые выполняются долго;

б) может потребоваться огромное общее количество функций, но для поиска решения требуется выполнить только малую их часть;

в) одновременно может запускаться большое количество функций.

Таким образом, можно находить управляющие решение для задач очень высокой сложности. Остается только создать способ обучения системы выработке нужного нам управляющего решения во всех возможных случаях.

Существует два принципиально разных способа создания (обучения) систем управления. Обозначим их как *классический* (алгоритмический) и *нейросетевой*. Рассмотрим их отдельно.

Классический подход

При классическом подходе предварительно создается модель управляемого устройства и среды, в которой он существует. Это значит, что формализуются все необходимые объекты, их свойства и законы их взаимодействия. На основе модели создаются правила принятия решения, которые так же приводятся к формализованному виду.

Предположим, что нам требуется реализовать промежуточную функцию g вида g: $\boldsymbol{P \rightarrow Q}$. Существует подмножество $P_1 \subset P$, такое что $\forall p \in P_1$ решение имеет вид q_1. При этом функция g нетривиальна, т.е. существует $p_2 \in P$, для которого решение равно q_2, причем $q_2 \neq q_1$. Чтобы разработать систему управления, требуется определить набор P_1: дать ему некоторый ключ (код/название), по которому он может быть найден в памяти, а также указать правило, по которому некоторое $p \in P_1$. Такой процесс называется *формализацией*,

а P_1 — *сущностью*, введенной в результате формализации.

На основе проведенной формализации все обрабатываемые объекты представляются в памяти УС кодами, уникальными для каждого объекта. Такие же коды получают свойства этих объектов. Успешная формализация позволяет создавать алгоритмы. Алгоритмы создают те правила, по которым будут работать промежуточные функции. Обмен между функциями происходит через ячейки памяти. Именно в этих ячейках функции находят значения аргументов, причем номера (адреса) ячеек так же передаются в виде аргумента. В результате своей работы функции либо меняют значения ячеек памяти, либо определяют, какие функции будут вызваны для дальнейшей обработки, т.е. происходит передача управления. [1, с. 7–32]

После формализации становится возможной программная реализация. В нашем случае реализация означает создание некоторой процедуры (может быть части процедуры или набора процедур) C. Такая процедура, получив на вход p, распознает сущность, а именно выдает соответствующее решение — q_1 или иное. Процедура должна иметь имя, список входных переменных, способ вызова (способ получения адреса точки входа в нее), участок кода, который будет производить распознавание. В общем случае одна процедура может распознавать сразу несколько сущностей.

Далее, для простоты изложения, будем считать, что преобразование алгоритмов в программный код вычислительной машины происходит идеальным образом, т.е. без ошибок: программный код всегда точно соответствует алгоритму.

Разработку устройства нельзя считать оконченной, пока есть данные, которые обрабатываются некорректно, т.е. принимается управляющее решение, которое нас не устраивает. В случае неправильной обработки требуется коррекция УС т.е. дообучение. При классическом подходе это означает, что должна быть изменена какая-либо из функций, либо добавлена новая. Следовательно, обучение системы потребует постоянного поиска функций, которые требуется изменить и значений, которым нужно указать другие реализующие функции. Из этого возникает задача следующего вида.

Задача

Дано: множество переменных и множество функций, связывающих эти переменные. Известно, что значение некоторых (входных) переменных Y_1 преобразуется в некоторое значение других (выходных) переменных X_1. Требуется найти подмножество функций, которое может быть изменено так, чтобы значение на выходе изменилось на X_2.

Сложность решения задачи зависит и не может быть меньше, чем NP. Это следует из того, что, если решение данной задачи с определенной сложностью будет найдено, это позволит получить такое же по сложности решение одной из NP-полных задач: выполнимость булевой функции. [3]

Таким образом, приходим к NP-сложности задачи обучения. Задачи подобной сложности на практике могут быть решены точно только для малого размера входа. Если размер входа большой, применяются эмпирические приближенные подходы. В нашем случае приближенный подход означает, что в качестве решения может предлагаться не точный код искомой функции, а других функций вместо нее. В задачах дискретной оптимизации приближенный подход имеет практический смысл, так как может использоваться любой из найденных вариантов, если он лучше, чем уже известные. Напротив, в случае обучения требуется получить точный код изменяемой (обучаемой) функции, так как даже минимальная ошибка в вычисленном коде/номере, вызовет обращение к принципиально другой функции.

Точно такие же рассуждения относятся к аргументам функций. Необходимость переобучения означает необходимость точного решения NP-трудной задачи поиска комбинации аргументов, обеспечивающих значение выхода.

Следовательно, надежное обучение сложных УС становится невозможным без быстрого точного решения NP-трудных задач, что является отдельной, возможно, принципиально неразрешимой проблемой.

Пока способа быстрого точно решения NP-трудных задач не существует, обучение при алгоритмическом подходе происходит следующим образом. Разработчики создают параллельную модель системы управления. Модель охватывает все сущности (все функции, аргументы и подмножества) управляемой системы, причем каждая сущность строго формализована, т.е. описана языком, понятным программисту. Программист в ходе корректировки системы должен в уме самостоятельно и корректно сформировать все модели. После этого он находит решение задачи, т.е. какие функции изменять и каким образом. Решение находится точно, но таким способом, который, с одной стороны, многократно описан, кажется очевидным и доступным каждому, с другой — не может быть строго формализован, а следовательно, автоматизирован.

Пока очевидно, что человек не может найти решение, просто зная условие задачи и просматривая

список сущностей — это слишком долгий процесс. Для эффективного поиска требуется дополнительная структура, связывающая между собой сущности таким способом, который позволит выполнить быстрый поиск необходимой сущности исходя из условий задачи. Такое решение приводит к существованию двух, тесно связанных между собой систем сущностей: одна для управления, другая для корректировки (обучения УС).

По мере усложнения УС, как поиск, так и необходимая структура становятся все сложнее, что начинает значительно замедлять разработку и добавление новых функций. [4, с. 38]

Так как на эффективное решение NP-трудных задач рассчитывать не приходится, можно пойти по пути поиска машины, которая будет допускать обучение в таких ситуациях. Рассмотрим работу УС на основе аналоговых преобразований. В ней не стоит проблема поиска номера, каждая из функций может выполнять следующие действия:

а) вызываться самостоятельно в момент изменения аргумента;
б) использовать только те ячейки, которые ей требуются;
в) интерпретировать ячейки любым необходимым способом.

Данные условия являются необходимыми, но недостаточными, формализуем их.

Условие 1. Машина не имеет памяти с общей (сквозной) нумерацией ячеек. Результаты выполнения функций записываются в некоторые ячейки, номера (или адреса, ключи) которых известны только тем функциям, которые будут обращаться к этим ячейкам.

Условие 2. Информация, которая используется в системе для принятия решений, не имеет ключа (кода) общего для всех функций. Информация формализована и закодирована, но интерпретировать ее могут только те функции, которые имеют к ней доступ.

Приведенные выше условия могут быть выполнены, если УС построена на основе нейросети. Оценим, как происходит реализация управляющей функции при таком подходе.

Нейросетевой подход

При нейросетевом подходе, алгоритмы могут оставаться неизменными для всех вариантов входных данных. В общем случае они могут быть двух видов: *обучающие* и *исполняющие*. Первые заполняют ячейки памяти на основе обучающих выборок, последние представляют собой возможные варианты входных данных и корректные ответы на них. Вторые формируют управляющие сигналы с учетом входных данных и содержимого памяти, заполненной в ходе обучения. Один исполняющий алгоритм после многократного выполнения над разными группами данных получает правильное управляющее решение.

Подобный подход привлекателен тем, что не требует какой-либо формализации, а значит, позволяет реализовать сложную задачу без построения параллельной модели системы управления. Кроме того, общее количество алгоритмов в такой структуре постоянно, т.е. не меняется по мере увеличения сложности структуры, а это упрощает и значительно ускоряет разработку.

Однако, у этого подхода есть фактор, ограничивающий его развитие. Прежде всего, для того чтобы обучить структуру выдавать правильные решения, требуется наличие соответствующих обучающих выборок (условие необходимое, но не достаточное). Так как не создаются модели, корректно описывающие поведение системы, обучающие выборки должны охватывать все возможные варианты входных данных (т.е. для всех сочетаний входов). Следовательно, размер выборки определяется разнообразием входных данных, а эта величина растет комбинаторно в зависимости от добавления в систему новых сущностей. В итоге, общий необходимый размер выборки будет определяться как экспоненциальная функция от размера входа. Для любых систем управления, кроме самых примитивных, общий необходимый объем выборки становится настолько огромным [5], что нет возможности не только проводить обучение, но даже хранить эти выборки в оперативном доступе.

Решением было бы поэтапное или поблочное обучение системы управления на выборках определенного размера. На такой выборке система обучается определенному ограниченному функционалу. Далее будет происходить усложнение системы через объединение блоков и дообучение по мере появления новых выборок, последние формируются с помощью моделирования и проведения экспериментов.

Такое решение при обычном нейросетевом подходе невозможно по следующей причине. Изменение структуры при обучении происходит без понимания, какие конкретно ячейки отвечают за каждое принятое решение. В результате попытка скорректировать какое-то решение через изменение некоторых ячеек может привести к неконтролируемому изменению тех решений, которые делаются правильно. Это приводит к тому, что надежная УС должна быть обучена сразу без последую-

щей коррекции, а значит, ее сложность ограничивается тем уровнем, который был доступен на первом этапе.

Чтобы допустить поэтапное развитие системы, необходима машина, которая выполняет следующее условие.

Условие 3. Введение новых функций (сущностей) должно происходить таким образом, чтобы не изменялись функции, уже существующие в системе.

Следующий ограничивающий фактор заключается в формировании управляющих выборок. Основной способ их формирования — эмпирический, т.е. эксперименты, тестирование и практический опыт. Даже при огромных затратах на эксперименты и обработку из-за комбинаторного усложнения эмпирически удается покрыть только малую долю всех состояний.

Остальные выборки должны формироваться теоретически на основе моделей в какой-то мере отражающих свойства реальных объектов. Модели позволяют получить формальные зависимости, далее на основе этих зависимостей можно автоматически формировать соответствующие выборки, получив в итоге то количество, которое требуется для реализации сложной УС.

Сейчас у такого способа есть главный недостаток: все модели, учитывающие свойства среды, в которой происходит управление, создаются классическим способом на основе формализаций и алгоритмов. При этом по сложности (количеству моделируемых сущностей) модели должны быть сравнимы с исходной средой. Это означает, что попытка реализовать необходимое количество моделей приведет к тем же проблемам, что и при реализации классическим способом. Эти проблемы следует исключить, поэтому вводится следующее условие.

Условие 4. Модели, имитирующие свойства среды, должны формироваться и далее корректироваться таким образом, чтобы выполнялись Условия 1, 2 и 3.

Заключение

С развитием информационных технологий и программного обеспечения создается впечатление, что при должных усилиях можно создать программное обеспечение, способное решить задачу управления системой любой сложности. Тем не менее практика показывает, что разработка надежных и сложных УС, работающих в открытой среде, достаточно быстро заходит в тупик.

По мнению авторов, проблема заключается не в недостатке ресурсов, а в принципиальной невозможности создать на детерминированной машине Тьюринга УС, поддерживающую большое число сущностей.

Необходимость выполнения приведенных выше условий является ограничителем развития современных систем, для которых требуется высокая надежность, точность выполнения. В результате такие системы вынуждены использовать крайне малое количество сущностей. По этой причине, например, на сегодняшний день наблюдается отсутствие заметного прогресса в функциональных возможностях роботов-пылесосов.

Попытка увеличить надежность, уменьшить частоту ошибок через увеличение количества обрабатываемых сущностей неминуемо приводит к экспоненциальному росту трудоемкости разработки. [6, с. 45]

Если будет разработана некоторая машина, которая сможет выполнять приведенные выше 4 условия, есть надежда, что она сможет решать задачи управления более сложные, нежели те, которые подвластны современным машинам.

ЛИТЕРАТУРА

1. Мирзоев М.С., Сатторов А. Э., Джонмахмадов И. Т. Математическая машина Тьюринга и вычислительная сложность. СПб: Прометей, 2020. — 88 с.
2. Мандель И. Д. Кластерный анализ. М.: Финансы и статистика, 1988. — 176 с.
3. Cook S. The complexity of theorem proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing. 1971, pp. 151–158.
4. Ураков А.Р., Тимеряев Т. В. Актуальные проблемы автоматического управления транспортными средствами. Интеллектуальные технологии на транспорте. 2021. № 1 (25). — С. 35–45.
5. Брюс П., Брюс Э., Гедек П. Практическая статистика для специалистов Data Science: Пер. с англ. СПб.: БХВ-Петербург, 2021. — 352 с.
6. Ураков А.Р., Федорова Г. И. Возможность частичной реализации при алгоритмическом подходе и большом списке сущностей. Системная инженерия и информационные технологии. 2022. Т. 4. № 2 (9). — С. 43–48.

© Ураков Айрат Ренатович (urakov@ufanet.ru), Поречный Сергей Сергеевич (porechny@mail.ru).

Журнал «Современная наука: актуальные проблемы теории и практики»