УДК 004.4 DOI: 10.26102/2310-6018/2019.27.4.021

РАЗРАБОТКА МЕТОДА САМОАДАПТАЦИИ ПРИКЛАДНОЙ ПРОГРАММНОЙ СИСТЕМЫ НА ОСНОВЕ ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ

А.М. Бершадский[1], А.С. Бождай[2], Ю.И. Евсеева[3], А.А. Гудков[4]
Пензенский государственный университет, Пенза, Российская Федерация
[1]e-mail: bam@pnzgu.ru
[2]e-mail: bozhday@yandex.ru
[3]e-mail: shymoda@mail.ru
[4]e-mail: alexei.gudkov@gmail.com

Резюме: В статье рассмотрены вопросы разработки метода самоадаптации прикладных программных систем на основе технологии машинного обучения. Рассмотрены различия между Model-Based и Model-Free подходами в обучении с подкреплением, обоснован выбор Model-Based подхода для создания метода самоадаптации программного обеспечения. Рассмотрено определение расширенного марковского процесса принятия решений, учитывающего роль ситуации в ходе самоадаптации программы. Предложена математическая модель пространства состояний программной системы, основанная на гиперграфовой формализации модели характеристик. На основе расширенного определения марковского процесса принятия решений, предложенной модели пространства состояний системы и концепции Model-Based подхода к машинному обучению с подкреплением разработан новый метод самоадаптации программного обеспечения, учитывающий влияние действий, производимых системой, на состояние окружающей среды. Приведен практический пример использования метода.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00408.

Ключевые слова: самоадаптивные программные системы, машинное обучение, обучение с подкреплением, искусственный интеллект.

Для цитирования: Бершадский А.М., Бождай А.С., Евсеева Ю.И., Гудков А.А. Разработка метода самоадаптации прикладной программной системы на основе технологии машинного обучения. *Моделирование, оптимизация и информационные технологии*. 2019;7(4):1-10. Доступно по: https://moit.vivt.ru/wp-content/uploads/2019/11/BershadskySoavtors_4_19_1.pdf DOI: 10.26102/2310-6018/2019.27.4.021

SOFTWARE SELF-ADAPTATION METHOD BASED ON MACHINE LEARNING TECHNOLOGY

A.M. Bershadsky, A.S. Bozhday, Y.I. Evseeva, A.A. Gudkov
Penza State University, Penza, Russia

Abstract: The article discusses development and application issues of software self-adaptation method based on machine learning technology. The differences between the Model-Based and Model-Free approaches in reinforcement learning are considered, the choice of the Model-Based approach for creating a software self-adaptation method is substantiated. The definition of an expanded Markov decision-making process that takes into account the role of the situation in the course of program self-adaptation is considered. A mathematical model of the state space of the software system is proposed, based on the hypergraphic formalization of the model of characteristics. Based on the expanded definition of the Markov decision-making process, the proposed model of the state space of the system, and the concept of the Model-Based approach to machine learning with reinforcement, a new method

of software self-adaptation was developed that takes into account the effect of the actions performed by the system on the state of the environment. A practical example of using the method is given.

The study was carried out with financial support of the Russian Foundation for Basic Research. Project № 18-07-00408

Keywords: self-adaptive software systems, machine learning, reinforcement learning, artificial intelligence.

For citation: Bershadsky A.M., Bozhday A.S., Evseeva Y.I., GudkoBelyakova A.A. Software self-adaptation method based on machine learning technology. *Modeling, optimization and information technology*. 2019;7(4): 1-10. Available by: https://moit.vivt.ru/wp-content/uploads/2019/11/BershadskySoavtors_4_19_1.pdf DOI: 10.26102/2310-6018/2019.27.4.021 (In Russ.).

Введение

Высокая сложность современных программных систем и большие ресурсозатраты на их разработку и сопровождение являются существенными проблемами современной программной инженерии. Проблема сложности программного обеспечения порождает проблему снижения качества его функционирования: заранее сложно учесть, как поведет себя система при различных внешних условиях, и на подобные оценки не всегда есть время. Программа, показывающая хорошие (например, в плане производительности) результаты в одних ситуациях, может недостаточно хорошо работать в других. Более того, разработка сложной программной системы, предусматривающей функционирование в различном программно-аппаратном окружении и при различных условиях (в том числе и пользовательских), может стать длительной и ресурсозатратной задачей.

Решением проблемы может стать наделение программного обеспечения способностью к самоадаптации.

Самоадаптивное программное обеспечение способно изменять собственное поведение в процессе выполнения с целью достижения основных системных целей. Непредсказуемые обстоятельства, такие как изменения в среде выполнения программы, системные сбои, возникновение новых требований и изменение приоритета старых, являются основными причинами, по которым программные системы вынуждены активировать свои адаптивные свойства. Чтобы эффективно справляться с возникающими в процессе функционирования неопределенностями, программа должна постоянно контролировать себя, собирать данные о собственном состоянии и состоянии среды выполнения, а затем анализировать полученные данные с целью определения ситуаций, в которых требуется запуск механизмов самоадаптации. Основную сложность в данном процессе представляет то, что система не просто должна вносить изменения в собственную структуру и поведение в процессе выполнения, но и делать это в соответствии с теми строгими требованиями, которые обычно предъявляются к программному обеспечению экспертами, разработчиками и пользователями. Проектирование подобных систем зачастую затруднительно, поскольку изначально сложно учесть все детали разработки системы. Поэтому более выгодным решением будет учет возникающих неопределенностей уже в процессе функционирования системы и последующая работа с ними. Реализовать подобный механизм возможно с помощью технологии машинного обучения - совокупности методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач.

Model-Free и Model-Based обучение: преимущества и недостатки при создании адаптивных программных систем

Обучение с подкреплением – это один из методов машинного обучения, в ходе которого испытуемая система, называемая также агентом, обучается, взаимодействуя со средой. Взаимодействие со средой обычно осуществляется на протяжении некоторого количества временных шагов. На каждом шаге агент наблюдает за состоянием среды и на основе этого наблюдения выбирает определенное действие. На следующем шаге агент получает вознаграждение (некоторое численное значение) за совершенное действие и переходит в новое состояние. Самодаптивная программная система в таком случае будет играть роль агента, наблюдая за текущим состоянием среды и используя для этого собственные модули мониторинга среды. Вознаграждение применительно к самоадаптивным программным системам обычно рассчитывается путем измерения различных значимых факторов и формирования функции полезности.

Методы обучения с подкреплением можно разделить на 2 класса: model-free и model-based. В случае обучением model-free значение имеет только вознаграждение - моделирование среды не производится. Одним из наиболее известных model-free методов является метод на основе модели actor-critic (исполнитель-критик). Метод предполагает использование двух нейросетей, одна из которых (actor) получает на вход состояние окружающей среды (state), а на выходе генерирует действия (actions), которые должны привести к увеличению вознаграждения (reward). Поначалу, пока не найдены какие-либо закономерности, действия выбираются случайным образом. Вторая нейросеть (critic) получает на вход состояние среды и действия, которые генерирует на выходе первая сеть, а на выходе также выдает значение награды, которая будет получена, если применить данные действия. В процессе функционирования системы нейросеть-критик обучается определять, какие именно действия приведут к увеличению награды при определенном состоянии среды, а знания, получаемые ею, можно использовать для генерации нейросетью-исполнителем наиболее оптимальных с точки зрения вознаграждения действий.

Model-based методы в корне отличаются от рассмотренного подхода. Нейросеть получает на вход комплекс ключевых параметров, описывающих состояние окружающей среды, и совершаемые действия, а на выходе генерирует возможные изменения среды (также может генерировать вознаграждение). Получаемое на выходе состояние среды подается на вход нейросети с новым набором действий. По сути, нейросеть в данном случае представляет собой модель предметной области, но не является системой принятия решений.

Преимуществом нейросетей в подходах Model-based является то, что они легко обучаются. Подобная особенность является следствием того, что для обучения сети можно использовать все возможные примеры поведения системы, а не только те, которые ведут к увеличению или уменьшению награды. Единственным недостатком подхода является ресурсоемкость модели - следствие необходимости учитывать реальную динамику исследуемой системы и среды.

В классическом обучении с подкреплением не учитывается одна из базовых концепции самоадаптаций программных систем - концепция ситуации [1]. Ситуация позволяет системе узнать, когда именно она должна следить за собственным состоянием и получать вознаграждение, а также предпринимать действие, соответствующее текущему состоянию. Ситуация предоставляет агенту информацию об окружающей среде таким образом, чтобы он вел себя так, как нужно при текущих параметрах среды. Агент может оставаться в одном и том же состоянии, но он должен предпринимать различные действия в различных ситуациях. Иными словами, ситуация накладывает

ограничения на множество доступных к выбору действий. В качестве примера можно рассмотреть суперкомпьютер, способный управлять распределением собственных ресурсов. В ситуации увеличения рабочей нагрузки следует увеличить количество ресурсов для повышения производительности. И напротив, количество ресурсов должно быть уменьшено для снижения энергопотребления при уменьшении рабочей нагрузки.

В основе предлагаемого метода самоадаптации программных систем лежит Model-based подход, дополненный определением ситуации. Такое техническое решение позволит программной системе адаптироваться к изменяющимся условиям внешней среды в реальном времени, что объясняется легкостью и высокой скоростью обучения Model-based нейросетей.

Метод самоадаптации программной системы на основе Model-Based обучения с подкреплением

Для формализации метода предлагается две математические модели:

– модель процесса принятия решений;

– модель пространства состояний системы.

Модель процесса принятия решений основана на расширенном понимании классического марковского процесса принятия решений, с учетом добавления в него фактора ситуации.

Пусть $I = \{i_1, i_2, ..., i_n\}$ - конечное множество ситуаций. Ситуация $i_j \in I$ определяет состояние среды или внутреннее событие, с которым может столкнуться агент, и запускает процесс самоадаптации. Тогда $A|_{s_j, i_j}$ указывает на множество возможных действий, которые агент может предпринять в состоянии s_j и ситуации i_j. Пусть $S*$ - декартово произведение множества состояний S и множества ситуаций I. Тогда марковский процесс принятия решений можно представить кортежем

$$M = \{S*, A, R, T\}, \quad (1)$$

где A – конечное множество действий, R – функция вознаграждения, получаемого системой при переходе из состояния $s \in S*$ в состояние $s' \in S*$ в результате действия $a \in A*$, T - функция, возвращающая вероятность того, что в результате действия $a \in A*$ система перейдет из состояния $s \in S*$ в состояние $s' \in S*$.

Для создания модели пространства состояний системы предлагается использовать модели изменчивости [2], в частности, их графическое представление - диаграммы характеристик (Рисунок 1).

Каждое отдельное состояние программной системы задается путем выбора определенного набора характеристик. Конкретное подмножество характеристик представляет собой конфигурацию модели. Следует отметить, что произвольные характеристики не могут составлять конфигурацию, а должны удовлетворять некоторым правилам исходной диаграммы.

Формализованное представление диаграммы характеристик будет иметь вид ориентированного гиперграфа [3,4]:

$$F = (N, E, \Delta, \Psi), \quad (2)$$

где $N = \{n_1, n_2, n_3, ..., n_n\}$ — конечное множество вершин, каждая из которых представляет собой характеристику диаграммы;

$E = \{E_1, E_2, E_3, ..., E_m\}, E_i \subseteq N \; \forall \; i = 1, ..., m$ — множество ребер гиперграфа, каждое из которых имеет собственное головное множество (множество вершин, в которые "входит" гиперребро) $H(E_i)$ мощностью q_i и хвостовое множество (множество вершин, из которых "исходит" гиперребро) $T(E_i)$ мощностью, равной 1;

$\Delta \in N$ — вершина, представляющая корневую характеристику диаграммы;

$\Psi : E \to M, M \subset N \times N$ — функция маркировки, которая присваивает значение мощности $mv(E_i) = (min, max) \in M$, где M — конечное множество значений мощности модели характеристик, N — конечное множество вершин гиперграфа, каждому ориентированному гиперребру E_i, так что $min, max \in Z \wedge min \geq 0, \; max > 0, min \leq max \wedge max \leq q_i$, где Z — множество целых чисел. Значение мощности — это пара из минимального и максимального количества вершин головного множества гиперребра, которые могут быть включены в конфигурацию диаграммы характеристик, которой соответствует гиперграф.

Рисунок 1 – Пример диаграммы характеристик

Поскольку пространство состояний системы предлагается определять с помощью модели характеристик, формула (2) будет также использоваться в качестве описания математической модели самоадаптивной программной системы. Тогда состояние системы можно понимать как подграф исходного графа *F*.

Предлагаемый метод самоадаптации программной системы, основанный на обучении с подкреплением, будет включать в себя следующие этапы:

1. Определение модели пространства состояний системы с помощью составления и формализации диаграммы характеристик.
2. Генерация множества $S*$ в соответствии с моделью пространства состояний.
3. Определение множества действий.
4. Основной цикл Model-Based обучения с подкреплением, включающий следующие этапы:
   a. действия в среде выполнения, сбор опыта (вознаграждений, получаемых системой за выполнение определенных действий в определенном состояний окружающей среды и при возникновении определенной ситуации);

b. вывод модели (зависимостей между конкретными действиями в определенных состояниях среды и ситуациях и вознаграждением);

c. обновление функции полезности в соответствии с выведенной моделью;

d. использование обновленной функции полезности для выбора действий, которые необходимо выполнить в дальнейшем;

e. перейти к следующей итерации цикла.

Основной цикл Model-Based обучения можно реализовать с помощью различных алгоритмов, к числу которых относятся, например, ряд алгоритмов Q-обучения [5], симуляционный поиск [6], простой поиск по методу Монте Карло [7] и метод на основе дерева поиска Монте Карло [8].

Рассмотрим более подробно Q-обучение. На основе получаемого от среды вознаграждения система формирует функцию полезности Q, что впоследствии дает возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. В обобщенном виде алгоритм Q-обучения включает следующие шаги:

1. Инициализация функции полезности $Q(s,a)$ и модели $Model(s,a)$ для каждого состояния $s \in S$ и действия $a \in A$. Функция $Model(s,a)$, принимая на вход текущее состояние и действие, возвращает следующее состояние и вознаграждение.

2. Бесконечный цикл:

a. Получение текущего нетерминального состояния s.

b. Получение текущего действия

$$a = ARGMAX(Q,s),$$

где $ARGMAX$ - функция, возвращающая действие, вознаграждение от выполнения которого в состоянии s максимально.

c. Выполнение действия a, получение вознаграждения r и нового состояния s'.

d. Обновление функции полезности:

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * MAX(Q,s') - Q(s,a)), \quad (3)$$

где α - фактор обучения (чем он выше, тем сильнее система доверяет новой информации), γ - фактор дисконтирования (чем он меньше, тем меньше система задумывается о выгоде от своих будущих действий), функция MAX определяет, какое максимальное вознаграждение возможно для текущего рассматриваемого состояния.

e. Обновление модели с использованием полученных значений r и s':

$$Model(s,a) \leftarrow r, s'.$$

f. Фаза планирования. Повторять n раз (число повторений n определяется разработчиком системы):

1. Выбор случайного состояния s из числа раннее наблюдаемых.
2. Выбор случайного действия a из числа ранее предпринятых в состоянии s.
3. Получение вознаграждения и нового состояния модели:

$$r, s' \leftarrow Model(s,a).$$

4. Обновление функции полезности в соответствии с формулой (3).

Примечательной особенностью приведенного в рамках метода алгоритма Q-обучения является фаза планирования. Именно планирование позволяет существенно ускорить процесс обучения, поскольку система использует уже не поступающие от внешнего мира данные, а собственную модель окружающей среды.

Пример использования метода

Чтобы проиллюстрировать пути практической реализации предложенного метода, рассмотрим тематическое исследование, основанное на технологии облачных вычислений. Рассмотрим случай, когда некоторая компания хочет развернуть свой сайт с помощью облачного сервиса. Сайт будет работать в двух режимах: графическом, который предоставляет больше информации и потребляет больше ресурсов, и текстовый. Назначение сайта заключается в том, чтобы предоставлять пользователю актуальную рыночную информацию с минимальными временными задержками. Владельцы сайта также заинтересованы в том, чтобы снизить затраты на приобретение облачных услуг путем минимизации избыточных в те или иные моменты времени ресурсов.

Очевидно, что такой веб-сайт является самоадаптируемой программной системой, причем наиболее важным фактором при планировании его поведения является время отклика. При анализе поставленной задачи обнаружены два возможных сценария поведения веб-сайта в случае, когда время отклика становится недопустимо большим:

– выделить больше ресурсов для повышения производительности сайта;

– переключиться на текстовый режим, чтобы сократить время вычислений.

Функция вознаграждения в данном случае будет связана с временем отклика сайта. Другими побочными факторами, связанными с вознаграждением, будут: режим работы (графический предпочтительнее), количество ресурсов (малое количество предпочтительнее). После выполнения данного шага уже можно составить первоначальную версию модели смены состояний.

Рассмотрим другую ситуацию - уменьшение трафика веб-сайта. Здесь возможны два сценария:

– снизить количество потребляемых ресурсов;

– переключиться в графический режим для предоставления пользователю более полной информации.

Метод самоадаптации применительно к данной задаче будет включать в себя следующие этапы работы:

1. Определение модели пространства состояний системы с помощью составления и формализации диаграммы характеристик. Диаграмма характеристик, соответствующая данному примеру, представлена на Рисунке 2.

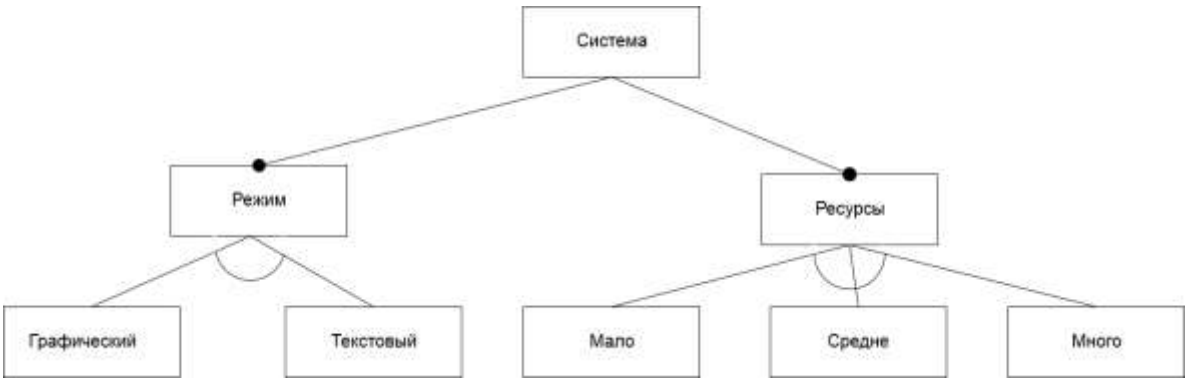

Рисунок 2 – Диаграмма характеристик, задающая пространство состояний самоадаптируемого веб-сервиса

2. Генерация множества состояний системы *S={{Режим:Графический, Ресурсы:Мало},{Режим:Графический, Ресурсы:Средне},{Режим:Графический,Ресурсы:Много},{Режим:Текстовый,*

Ресурсы:Мало},{Режим:Текстовый, Ресурсы:Средне},{Режим:Текстовый, Ресурсы:Много}} в соответствии с диаграммой характеристик на Рисунке 2. Определение множества ситуаций *I={ВремяОтклика:Нормальное, ВремяОтклика:Низкое}*. Генерация множества $S^* = S \times I$. На Рисунке 3 элементы S^* изображены вершинами графа переходов.

3. Определение множества действий *A={ДобавитьРесурс, ОсвободитьРесурс, ПереключитьРежимВГрафический, ПереключитьРежимВТекстовый }.*

4. Основной цикл Model-Based обучения с подкреплением, реализованный с помощью алгоритма Q-обучения, в котором в качестве вознаграждения *r* выступает величина, обратная времени отклика.

Диаграмма переходов между состояниями облачного самоадаптируемого веб-сервиса приведена на Рисунке 3. Стрелки на Рисунке обозначают действия, переводящие систему из одного состояния в другое. Каждому действию соответствует свое вознаграждение, которое вычисляется при выполнении алгоритма обучения.

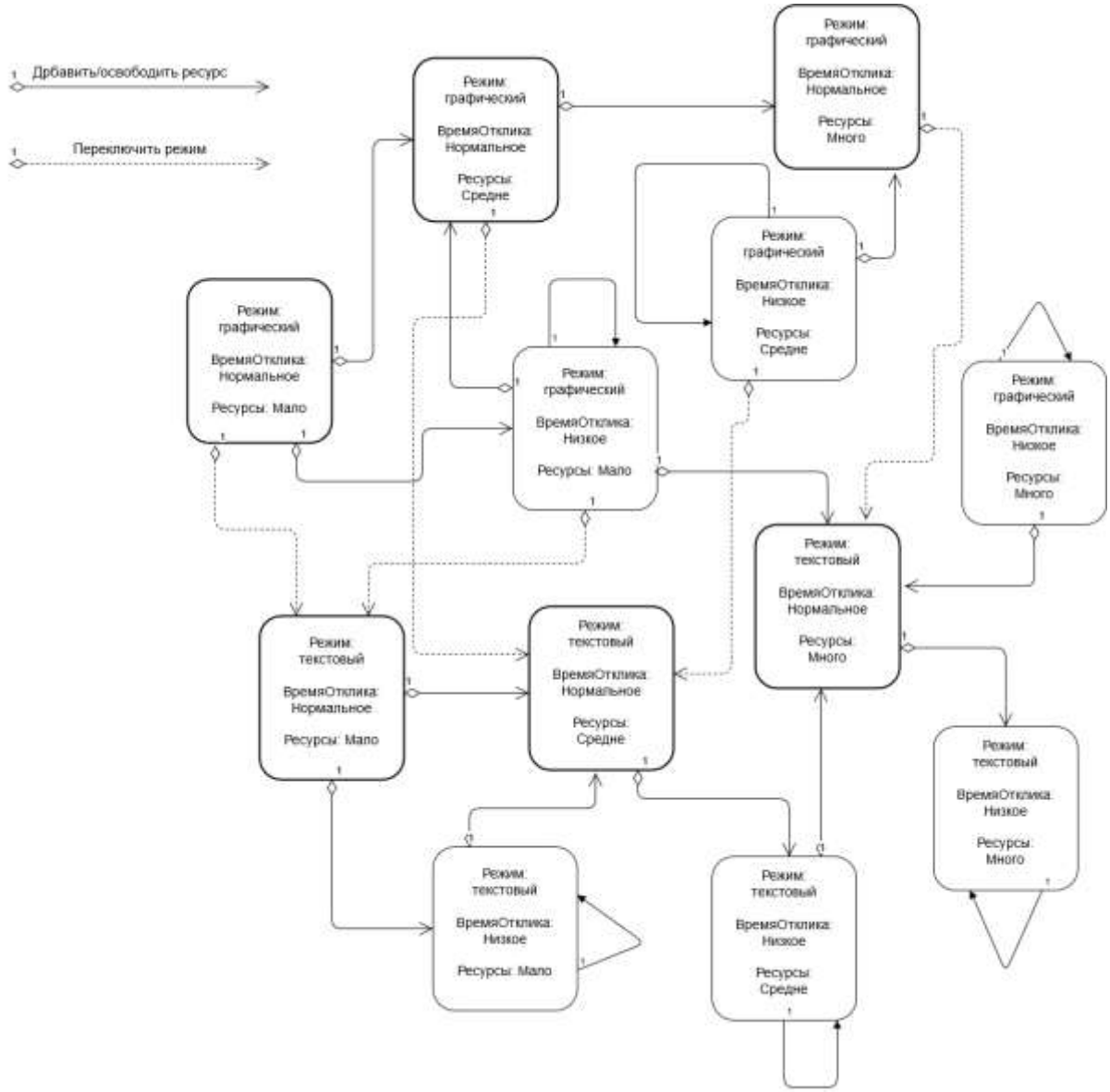

Рисунок 3 – Диаграмма переходов между состояниями самоадаптируемого веб-сервиса

В результате обучения система учится оптимально реагировать на изменение внешних условий (в результате которых происходит увеличение/уменьшение времени отклика) и накапливает знания о том, как нужно изменить свое состояние, чтобы достичь максимального вознаграждения.

Заключение

В статье рассмотрены преимущества использования технологии машинного обучения с подкреплением при создании самоадаптивных программных систем. Предложен новый метод самоадаптации программных систем, основанный на совместном использовании технологии машинного обучения с подкреплением (Model-based подход) и технологии моделирования изменчивости. Отличительной особенностью рассмотренного метода является возможность эффективной самоадаптации программного обеспечения к внешним условиям в режиме реального времени. Это позволит программным системам оперативно реагировать на непредвиденные обстоятельства внешней среды, к числу которых относятся, например, сбои в работе аппаратуры. Программы, создаваемые на основе предложенного метода, не ограничиваются какой-либо узкой областью применения: это могут быть как обучающие видеоигры, так и обеспечение сложных вычислительных комплексов.

ЛИТЕРАТУРА

1. Han H. Model-based Reinforcement Learning Approach for Planning in Self-Adaptive System. *International Conference on Ubiquitous Information Management and Communication*. New Jersey: IEEE; 2015:156-178.
2. Simmonds J., Bastarrica M.C. Modeling variability in software process lines. *Departamento de Ciencias de la Computación*. 2011;4:93-100.
3. Бершадский А.М., Бождай А.С., Евсеева Ю.И., Гудков А.А. Математическая модель рефлексии самоадаптивных программных систем. *Известия Волгоградского государственного технического университета*. 2018;8:7-14.
4. Бождай А.С., Евсеева Ю.И. Метод рефлексивной самоадаптации программных систем. *Известия высших учебных заведений. Поволжский регион. Технические науки*. 2018;46(2):74-86.
5. Machine Learning Proceedings 1991: Proceedings of the Eighth International Workshop (ML91). Elsevier Science; 2014.
6. Silver D. Reinforcement learning and simulation-based search in computer go. Alta., Canada: University of Alberta Edmonton; 2009.
7. Fishman, George S. Monte Carlo: concepts, algorithms, and applications. Springer; 1996.
8. Colum R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Computers and Games, 5th International Conference*. Turin, Italy: Springer; 2006.

REFERENCES

1. Han H. Model-based Reinforcement Learning Approach for Planning in Self-Adaptive System. *International Conference on Ubiquitous Information Management and Communication*. New Jersey: IEEE; 2015:156-178.
2. Simmonds J., Bastarrica M.C. Modeling variability in software process lines. *Departamento de Ciencias de la Computación*. 2011;4:93-100.

3. Bershadskij A.M., Bozhdaj A.S., Evseeva YU.I., Gudkov A.A. Matematicheskaya model refleksii samoadaptivnyh programmnyh system. *Izvestiya Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta*. 2018;8:7-14.
4. Bozhdaj A.S., Evseeva YU.I. Metod refleksivnoj samoadaptacii programmnyh system. *Izvestiya vysshih uchebnyh zavedenij. Povolzhskij region. Tekhnicheskie nauki*. 2018;46(2): 74-86.
5. Machine Learning Proceedings 1991: Proceedings of the Eighth International Workshop (ML91). Elsevier Science; 2014.
6. Silver D. Reinforcement learning and simulation-based search in computer go. Alta., Canada: University of Alberta Edmonton; 2009.
7. Fishman, George S. Monte Carlo: concepts, algorithms, and applications. Springer; 1996.
8. Colum R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Computers and Games, 5th International Conference*. Turin, Italy: Springer; 2006.

ИНФОРМАЦИЯОБАВТОРЕ / INFORMATIONS ABOUT AUTHORS

Бершадский Александр Моисеевич, д.т.н., профессор, заведующий кафедрой САПР Пензенского государственного университета
ORCID: 0000-0001-9467-4206

Alexander M. Bershadsky, Doctor of Technical Sciences, Professor, Head of CAD Department of Penza State University

Бождай Александр Сергеевич, д.т.н., профессор кафедры САПР Пензенского государственного университета
ORCID: 0000-0003-0677-7673

Alexander S. Bozhday, Doctor of Technical Sciences, Professor, Department of CAD, Penza State University

Евсеева Юлия Игоревна, к.т.н., доцент кафедры САПР Пензенского государственного университета

Julia I. Evseeva, Candidate of Engineering, Associate Professor, Department of CAD, Penza State University

Гудков Алексей Анатольевич, к.т.н., доцент кафедры САПР Пензенского государственного университета

Alexey A. Gudkov, Ph.D., Associate Professor, Department of CAD, Penza State University