Санкт-Петербургский государственный университет

УЖВА Денис Романович

Выпускная квалификационная работа

Мультиагентные технологии в киберфизических системах

Уровень образования: магистратура

Направление *02.04.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *ВМ.5665.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
д.ф.-м.н., профессор кафедры системного программирования, О.Н. Граничин

Рецензент:
старший инженер ООО «Техкомпания Хуавэй» Н.В. Устюжанин

Санкт-Петербург
2021

Saint Petersburg State University

Denis Uzhva

Master's Thesis

Multiagent Technology in Cyber-Physical Systems

Education level: master

Speciality *02.04.03 "Software and Administration of Information Systems"*

Programme *BM.5665.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
Sc.D, prof. O.N. Granichin

Reviewer:
Senior Engineer at "Huawei Technologies Ltd." N.V. Ustiuzhanin

Saint Petersburg
2021

Оглавление

Введение

В последнее время вычислительные устройства претерпевают значительные изменения ввиду возрастающей интеллектуализации всевозможных областей человеческой деятельности, будь то производство, медицина или, например, логистика. Для решения крупномасштабных задач требуется создавать сложные информационные системы с большим количеством взаимодействующих между собой компьютеров, поскольку текущие задачи не представляется возможным решить ресурсами одной машины.

При этом информационные процессы становятся всё более объёмными — возникает проблема так называемых больших данных (Big Data), работа с которыми возможна лишь объединением усилий многих вычислительных устройств в единую сеть для распределённой совместной работы. На сегодняшний день большие данные возникают во многих областях человеческой деятельности: например, в медицине приходится иметь дело с анализом трёхмерных снимков головного мозга и других органов [3, 6, 63], в сельском хозяйстве также присутствует потребность в оптимальной координации работы большого числа сельскохозяйственной техники [49], а в физике элементарных частиц задача обработки данных в экспериментах по столкновению ионов, объём которых исчисляется петабайтами, единовременно решается крупномасштабной сетью компьютеров, распределённых по всему миру [40, 41]. Стоит отметить, что распределённые вычисления имеют ряд полезных особенностей: масштабируемость (решение проблемы можно ускорить аддитивным внедрением новых устройств в общую сеть), отказоустойчивость (при выходе из строя нескольких устройств задачи, возложенные на них, перераспределяются на другие компьютеры) и безопасность (атака на некоторые вычислительные узлы не выведет из строя всю систему, поскольку от таких узлов можно будет временно избавиться без существенного ущерба для работы всей системы в целом). Эти особенности порождают всё больше актуальных исследований, направленных на создание технологий распределённого анализа больших данных.

Проблема больших данных характерна не только для информационных систем в классическом понимании как сети из серверов и клиентов. Существует другая тенденция: растет количество всевозможных встроенных устройств, т.е. компьютеры становятся своеобразным "мозгом" того или иного физического объекта. Объединённые в сеть, "умные" объекты образуют то, что сегодня называют "Интернетом вещей" — интернетом киберфизических (от англ. cyber-physical systems) объектов. Так, например, сегодня всё более активно происходит разработка беспилотных автомобилей, что открывает пространство для новых решений задачи "балансировки" транспортного трафика на дорогах. Эти решения можно разбить на два класса: централизованные и децентрализованные [50]. В первом случае автомобили сообщают свои координаты некоторому дата-центру, играющему роль координатора, который принимает решения по балансировке всей системы. Во втором — такой центр отсутствует, и машины с помощью локальных взаимодействий приходят к консенсусу, удовлетворяющему решению поставленной задачи. На сегодняшний день централизованные иерархические стратегии более предпочтительны, поскольку обеспечивают прямой и понятный мониторинг системы, что важно для стабильной эксплуатации. Однако простота даётся ценой неизбежных затрат на коммуникации между компьютерами и дата-центрами, что в больших масштабах может привести к нехватке пропускной способности каналов коммуникаций [16, 47]. Иначе говоря, естественным образом ограниченная пропускная способность каналов ограничивает и масштабируемость системы, что делает такое решение не пригодным для работы с большими данными. В то же время описанный недостаток практически отсутствует в децентрализованной стратегии: в зависимости от нагруженности новые узлы можно присоединить к тем компьютерам, что имеют сравнительно мало соединений с другими. При этом не возникает необходимости соединять все узлы в полносвязную сеть, поскольку, влияя на состояние соседей (здесь и далее под соседями имеются в виду коммуницирующие друг с другом узлы), так или иначе каждый компьютер в системе опосредованно будет связан с остальными. Такие системы

принято называть мультиагентными, а узлы в таком случае называются (интеллектуальными) агентами [17, 23, 62]. Под мультиагентными технологиями же, соответственно, следует понимать решения сложных задач децентрализованным способом, при котором решение (зачастую спонтанно) возникает в процессе взаимодействий агентов в системе.

На сегодняшний день мультиагентные системы (МАС) пока ещё не получили широкого распространения ввиду их недостаточной изученности. Однако такие многоагентные алгоритмы, как метод роя частиц (Particle Swarm Optimization, PSO) [29, 60], гравитационного поиска [28, 53], стохастический диффузный поиск [1] или муравьиный алгоритм [2] уже успешно применяются в индустрии для решения задач оптимизации. Основным препятствием при разработке таких алгоритмов служит эмерджентность результата, заключающаяся в несводимости поведения отдельных агентов к поведению целой системы [22, 37]. В связи с этим сегодня мы пока находимся на стадии рассмотрения относительно простых задач о достижении консенсусных состояний, например, о балансировке загруженности вычислительных узлов [9, 12]. Однако нарастающая потребность в децентрализованных решениях и, как следствие, повышающийся интерес к мультиагентным технологиям неизбежно приводят к обсуждению проектов сложных роевых робототехнических систем для медицины и индустрии [48].

В первую очередь мультиагентные системы являются объектом изучения компьютерных наук, поскольку мультиагентные технологии позволяют эффективно решать комбинаторные задачи. Теория мультиагентного управления исследуется кибернетикой, внёсшей большой вклад в развитие математического аппарата динамических систем, моделируемых при помощи дифференциальных уравнений [18]. Основным альтернативным подходом к формализации мультиагентных систем является моделирование при помощи автоматов [36, 54]. Однако для моделирования именно киберфизических систем первый подход является наиболее удобным, поскольку изменение состояний киберфизических агентов (например, беспилотных летательных аппаратов) происходит в физическом мире по законам механики.

Среди наиболее развитых методов управления МАС можно выделить два подхода: глобальное управление, при котором каждому агенту задают одинаковый алгоритм поведения, не зависящий от действий самих агентов [61], и локальное — при котором у каждого агента свой собственный алгоритм взаимодействия с остальными, зависящий от состояния соседних агентов [18, 25, 26]. В первом случае получается довольно простая для понимания и, что более важно, для эксплуатации схема управления. Однако для простоты приходится жертвовать вариативностью множества достижимых состояний. Теория локального управления на сегодняшний день не так хорошо развита, как в случае с глобальным управлением, но имеет большой потенциал из-за большой вариативности возможных технологических решений [18]. Несмотря на заманчивые перспективы, у локального управления есть большая проблема: в крупномасштабных системах при необходимости изменить стратегию управления (например, при адаптации к новой дороге в задаче балансировки транспортного трафика) придётся "точечно" обратиться к каждому агенту индивидуально, что повлечёт большие коммуникационные и вычислительные затраты со стороны дата-центра.

В последнее время растёт популярность исследований явления кластеризации в МАС [25, 26, 39]. Процесс кластеризации может происходить спонтанно: если индивидуальные цели агентов совпадают, то в совокупности они могут сформировать некую структуру (подсистему), воспринимаемую как целое. Спонтанная группировка является весьма распространённым явлением как в природе (объединение животных в стада [35], синхронная работа участков головного мозга [52]), так и в человеческой и интеллектуальной деятельности (объединение поселений в государства, серверов в кластеры, обнаружение общих закономерностей в данных [10]).

Подобно тому, как кластеризация происходит в природе, это явление также характерно для групп роботов. В действительности кластеризацию можно использовать для управления целыми группами агентов — при этом децентрализация переходит из уровня индивидуальных агентов на уровень взаимодействующих кластеров. При большом коли-

честве агентов в системе проследить всё разнообразие взаимодействий между ними довольно сложно, в то же время кластеризация позволяет эффективно группировать агентов, что позволяет снизить количество управляющих воздействий и, таким образом, упростить систему. Более того, поскольку отдельные группы агентов обособляются от других, их траектории в пространстве состояний имеют разреженное представление. Это позволяет применять методы сжатия данных при передаче информации о МАС в дата-центр, например, для принятия решений об изменение локальной и кластерной стратегии управления.

Таким образом, актуальность парадигмы кластерного управления подкрепляется многочисленными упомянутыми выше примерами востребованности в индустрии и науке. При использовании кластерного управления по внешним удалённым наблюдениям за поведением агентов определяются группы агентов, воспринимаемые управляющим дата-центром как единое целое. В связи с этим возникает мотивация разработать новый метод сжатия больших данных, способный снизить нагрузку на канал обмена данными между дата-центром и агентами без существенных потерь информации о состоянии МАС.

Постановка задачи

Целью настоящей работы является создание алгоритма и разработка прототипа программного обеспечения для кластерного управления крупномасштабными киберфизическими мультиагентными системами. Для достижения цели в рамках работы были сформулированы следующие задачи.

1. Проанализировать методы управления мультиагентными киберфизическими системами.

2. Разработать математический фреймворк для описания мультиагентных систем, формализующий явление “кластерных потоков”; проанализировать с его помощью модель осцилляторов Курамото как пример киберфизической мультиагентной системы.

3. Разработать алгоритм обработки данных о состоянии агентов в мультиагентной системе с последующим синтезом управления кластерами в системе.

4. Сформулировать требования к программному обеспечению для кластерного управления мультиагентными системами.

5. Реализовать прототип программного обеспечения для кластерного управления мультиагентными системами.

6. Выполнить апробацию алгоритма на примере модели осцилляторов Курамото.

1. Мультиагентное управление

Целью обзора методов управления МАС является сравнение существующих методов для объяснения мотивации нового алгоритма, представляемого далее в настоящей работе.

1.1. Постановка задачи управления МАС

Управление мультиагентной системой предполагает некоторый перечень манипуляций над ней, приводящих текущее состояние системы к некоторому заранее заданному желаемому виду. Однако, прежде чем формулировать задачу управления, следует разобраться с тем, что такое агент и чем он отличается от объекта — ведь и то, и другое являются структурными единицами системы. Итак, если следовать определениям в [17, 25, 26, 46, 62], агентом называется адаптивный автономный элемент системы. Кроме того, в классической книге Вулдриджа [62] по мультиагентным системам акцентируется внимание на следующих трёх характеристиках агента, отличающих его от объекта:

- реактивность — агент своевременно реагирует на события во внешней среде, в которой существует;
- проактивность — агент имеет модель поведения, основывающуюся на определённой для него индивидуальной цели;
- социальность — агент способен выделять других агентов во внешней среде и взаимодействовать с ними для помощи себе в достижении своей цели.

Совместное поведение агентов приводит к обсуждаемому ранее эмерджентному поведению целой системы, из-за чего при мультиагентном управлении не преследуется цель определить явные зависимости между действиями агента и действиями системы. Вместо этого МАС рассматривается как “чёрный ящик”, на который можно воздействовать лишь через строго обозначенные “входы”, а также наблюдать результат через

“выходы”. К примеру, в задаче мультиагентной балансировки нагрузки на вычислительные узлы примерами таких входов и выходов могут послужить сетевые порты, через которые опосредованно при помощи алгоритмов в самих узлах производится управление. Таким образом, задача управления МАС формулируется как отыскивание таких входных воздействий на систему, которые приводят её к желаемому состоянию, верифицируемому через её выходы.

1.2. Моделирование мультиагентных систем

Для формализации поведения мультиагентных систем и соответствующих управляющий воздействий требуется отразить динамику состояний агентов. Выделяют два основных подхода к моделированию МАС:

1. Моделирование при помощи автоматов [36, 54] — агенту сопоставляется автомат, при этом состояние агента соответствует состоянию автомата.

2. Моделирование с использованием аппарата динамических систем в виде систем дифференциальных уравнений [18,19,25,26,46] — состояние агента описывается дифференцируемой функцией от времени, а изменение этого состояния является функцией текущего состояния агента, состояний агентов-соседей, а также зависит от воздействий со стороны внешней среды.

В первом случае поведение МАС как сеть автоматов можно описать, например, разностным уравнением авторегрессионной модели:

$$X_t = c + \sum_{k=1}^{p} \varphi_k X_{t-k} + \eta_t, \tag{1}$$

где X_t — состояние агента в момент времени t, c — константа, $\varphi_1, \ldots, \varphi_p$ — параметры, а η_t — стохастическая добавка (возникающая из-за неизвестных помех). При $p = 1$ получается так называемый Марковский процесс, подчиняющийся принципу локальности по времени, из-за чего

подвержен влиянию эффекта “отсутствия памяти”. В модель (1) можно добавить связность между агентами при помощи добавления зависимости состояния агента X от состояний соседних агентов. Помимо разностных уравнений для описания МАС используют вероятностные модели, основанные на формализме цепей Маркова [36]. Такой подход удобен для описания стохастических дискретных процессов, что актуально в контексте крупномасштабных систем с большим числом цифровых агентов.

Несмотря на итеративный подход автоматного моделирования и, соответственно, простоту реализации модели на классическом компьютере, киберфизические системы изначально подчиняются законам механики. В этом случае удобнее использовать аппарат динамических систем, с помощью которого динамика состояния агента (в сети из N агентов) описывается как

$$\dot{x_i}(t) = f_i(x_i(t), u_i(t), \eta_i(t)), \tag{2}$$

где $x_i(t) \in \mathbb{R}^{n_i}$ является вектором состояния агента $i \in \mathcal{N} = \{1, \ldots, N\}$, $u_i(t)$ — управляющее воздействие, $\eta_i(t) \in \mathbb{R}^{m_i}$ — стохастическая добавка. Всюду далее рассматривается модель (2) и её модификации.

1.3. Типы целей управления

В кибернетике и теории управления принято предварительно задавать цель управления, определяющую вид функции $u_i(t)$ в (2). Согласно классификации, предлагаемой в [18], можно выделить следующие цели:

1. Регулирование (стабилизация) — приведение состояния системы к заранее заданному константному значению x_*:

$$\lim_{x \to +\infty} x(t) = x_*.$$

2. Слежение — приведение динамики состояния системы к некоторой функции $x_*(t)$:

$$\lim_{x \to +\infty} |x(t) - x_*(t)| = 0.$$

3. Возбуждение колебаний — частный случай цели слежения, формализуемый скалярной целевой функцией $G(x)$:

$$\lim_{x \to +\infty} G(x(t)) = G_*.$$

4. Модификация предельных множеств (аттракторов) — широкий класс целей, для которых характерны качественные изменения состояния системы: например, изменение положения точки бифуркации в фазовом пространстве или преобразование типа равновесия.

5. Синхронизация — сближение переменных состояния двух агентов в одной системе (или двух одноагентных систем):

$$\lim_{x \to +\infty} |x_i(t) - x_j(t)| = 0.$$

Выделенные классы целей первоначально предназначались для одноагентных систем с одной общей функцией состояния. Можно заметить, что межагентное взаимодействие предполагается только в цели синхронизации. Именно по этой причине задачи синхронизации на сегодняшний день особенно актуальны в контексте мультиагентных технологий.

Цели управления также можно отнести к одной из двух категорий:

1. Глобальная цель [61] — одинакова для всех агентов, выражается в одинаковости воздействия на всех агентов системы.

2. Локальная цель [18, 46] — воздействие определяется локальными событиями в системе, сообщения о которых поступают к агентам с помощью коммуникации с соседними.

По своей сути цель синхронизации является локальной, поэтому соответствующие управляющие воздействия могут зависеть как от состояния самого агента, так и его соседей. Однако гибридные цели со сложными управляющими воздействиями, сочетающими локальные и глобальные цели, также возможны.

Если рассматривать глобальное управление как частный (вырожденный) случай локального, то возникает вопрос исследования промежуточных способов управления. В действительности динамика крупномасштабных систем, таких как группы роботов, работающих в постоянно меняющейся среде, зачастую слишком сложна, чтобы её можно было контролировать при помощи локально-глобальных целей. Однако иногда агенты популяции демонстрируют разделение на группы — кластеры: агенты одной группы синхронизируются, в то время как синхронизация между агентами из разных кластеров отсутствует. Благодаря кластерной самоорганизации возможно формирование стратегий управления, в которых кластеры считаются элементарными единицами системы. Это позволяет уменьшить необходимое количество управляющих воздействий, упрощая решение задачи достижения определённого состояния в системе пропорционально отношению количества кластеров к количеству агентов. Зачастую количество кластеров оказывается гораздо меньше количества составляющих его агентов, в связи с чем описанное упрощение является значительным. Например, в организме человека количество органов на порядки меньше количества составляющих их клеток.

Описанная новая парадигма моделирования МАС называется кластерным управлением, “заполняющим” нишу между локальным и глобальным. Таким образом, можно определить три уровня целей управления мультиагентными системами: микроскопический (уровень отдельных агентов), мезоскопический (уровень кластеров) и макроскопический (система в целом). Описанное отношение для наглядности можно записать следующим уравнением

$$N \gg m \gg 1,$$

где N — число агентов в системе и m — число кластеров. Описываемый в следующей главе фреймворк формализует понятие кластерной синхронизации для разрабатываемого алгоритма кластерного управления. В свою очередь, это понятие важно для понимания явления *кластерных потоков* — изменяющейся во времени кластеризации агентов.

1.4. Методика опознания по сжатию

Совокупный объём данных о состояниях агентов, обмен которыми происходит в крупномасштабных мультиагентных системах, оказывается колоссальным с точки зрения моделирования этих систем и в особенности выбора стратегии управления. Метод опознания по сжатию (Compressive Sensing, CS) [4, 7, 14, 24] позволяет сжимать большие потоковые данные по мере получения с помощью их частичного суммирования. В контексте киберфизических систем это могут быть, например, телеметрические данные роя беспилотных летательных аппаратов, собираемые дата-центром для синтеза кластерного управления.

Сама возможность сжимать данные о МАС с последующей реконструкцией информации с минимальными потерями появляется при наличии кластерной синхронизации. В связи с этим далее будут рассматриваться только такие системы, в которых агенты формируют группы. Поскольку кластеры занимают сравнительно небольшую область в пространстве состояний, то целые группы агентов в них можно рассматривать как единые элементы, управлять которыми можно как самостоятельными элементарными единицами системы. При наблюдении за МАС с кластеризацией её пространство состояний имеет разреженное представление, которое, согласно теории о Compressive Sensing, можно подвергать сжатию как разреженный сигнал.

Пусть сигнал $f \in \mathbb{R}^N$ является s-разреженным в некотором базисе $\Psi \in \mathbb{R}^{N \times N}$, т.е. $f = \Psi x$, где у вектора x не более s ($s \ll N$) ненулевых элементов. Это означает, что в некотором базисе исчерпывающие данные о закодированной информации содержатся только в s из N элементов массива данных. Метод опознания по сжатию предлагает следующую схему компрессии:

$$y = \Phi\Psi x = Ax, \tag{3}$$

где Φ — $m \times N (m \ll N)$ т.н. “sampling matrix”, A — матрица измерений, а $y \in \mathbb{R}^m$ — вектор сжатых наблюдений. Поскольку $m \ll N$, то задача оценки x при данном y (реконструкции, декомпрессии) имеет

бесконечно много решений. Однако, если x является s-разреженным, то реконструкция становится возможной без существенных потерь значимых данных, при этом достаточно, чтобы $m \sim s \log(N/s)$ [24]. В соответствии с [4, 7, 14], среди решений (3) предпочтительны те, что максимизируют количество нулевых элементов в оценке x.

В общем случае для этого требуется решить задачу ℓ_0-оптимизации, поскольку она позволяет отыскать наиболее разреженное решение, однако такая задача является NP-сложной. Однако ℓ_1-оптимальные оценки оценки $\hat{x}$ исходного вектора x хорошо аппроксимируют исходный вектор состояний, при этом задача ℓ_1-оптимизации (линейного программирования) решается за полиномиальное время [7, 51]. Такой скорости сходимости можно добиться при помощи одного из двух представленных ниже популярных подходов:

1) метод внутренней точки [44, 45];

2) симплекс-алгоритм [15, 27].

Как показано в [59], метод внутренней точки сходится за 10–100 итераций, в то время как альтернативному представленному выше алгоритму требуется $2n$–$3n$ итераций для сходимости, где n в случае МАС трактуется как размерность пространства состояний системы. Таким образом, метод внутренней точки используется в алгоритме кластерного управления с компрессией для мультиагентных систем, поскольку лучше подходит для крупномасштабных МАС большой размерности.

В качестве матрицы A, в соответствии со свойством ограниченной изометрии (Restricted Isometry Property, RIP) [24], можно выбрать матрицу с рандомизированными элементами, поскольку в этом случае её столбцы будут иметь наименьшую корреляцию с любым базисом, что обеспечит наилучшее “кодирование”.

В [24] показано, что для реконструкции x достаточно лишь $m = c_1 s \log(N/s)$ измерений. На практике же зачастую работает следующая эвристика: $m \approx 4s$.

В качестве альтернативы можно рассмотреть генерацию матрицы A при помощи методов машинного обучения [34]. Однако в этом слу-

чае условие минимума инкогеренции будет соблюдаться не для всякого вектора состояния системы, в отличие от рандомизированного подхода. Это приведёт к тому, что для каждой новой стратегии управления, приводящей к новым паттернам в пространстве состояний, придётся генерировать новую матрицу измерений. Ценой несущественного возможного усиления компрессии (поскольку для специализированной матрицы A можно будет уменьшить размерность y) существенно замедлится работа всего алгоритма кластерного управления с компрессией, поскольку в процесс включится элемент машинного обучения, требующий много времени и ресурсов для подготовки новой матрицы.

1.5. Мультиагентная модель осцилляторов Курамото

В качестве примера с помощью описываемого далее фреймворка анализируется нелинейная модель Курамото [33], в которой возникает синхронизация агентов-осцилляторов. Её динамика для некоторой сети из N агентов с одной степенью свободы (которую зачастую называют фазой осциллятора) описывается следующей системой дифференциальных уравнений:

$$\dot{\theta}_i(t) = w_i + \sum_{j=1}^{N} K_{ij} \sin\left(\theta_j(t) - \theta_i(t)\right), \tag{4}$$

где $\theta_i(t)$ — фаза агента i, K_{ij} — взвешенная матрица связности сети, а w_i — собственная частота. В соответствии с [5], [8] и [30], агенты приходят в состояние частотной ($\dot{\theta}_i = \dot{\theta}_j \ \forall i, j \in \mathcal{N}$) или фазовой ($\theta_i = \theta_j \ \forall i, j \in \{1, \ldots, N\}$) синхронизации при некоторых условиях на w_i и K_{ij}.

Существует множество расширений модели Курамото, например, модель с зависящими от времени матрицей связности $K_{ij}(t)$ и частотами $w_i(t)$ [38]. Известно также, что была изучена модель с задержками по времени и фазе [32, 42]. Кроме того, модель Курамото имеет мультиплексную версию [52], были также попытки представить её как квантовую систему [58]. В статье [57] обсуждалась кластерная синхро-

низация при некоторых условиях на параметры.

Также модель Курамото способна адекватно описывать работу коры головного мозга человека [52]. Так, было выявлено три режима работы мозга, соответствующих трем состояниям сети осцилляторов: асинхронному, полностью синхронному и "хаотическому'', соответствующему кластерной синхронизации. Стоит отметить, что эта модель также подходит для описания разнообразных биологических роевых систем [33].

1.6. Алгоритмы поиска кластеров применительно к MAC

Задача кластеризации предполагает разбиение некоторой совокупности на (обособленные) группы. В контексте MAC эта задача эквивалентна задаче поиска групп в популяции агентов системы.

Все алгоритмы кластеризации можно разделить на четыре вида:

- иерархические [21, 31];
- центроидные [43, 55];
- кластеризация по распределению [13];
- кластеризация по плотности [11, 20].

В иерархическом подходе формируется кластерное дерево, листья которого соответствуют индивидуальным образцам данных из выборки, а корень — всей выборке. При этом ветви такого дерева представляют из себя кластеры, разветвляющиеся на более мелкие вплоть до уровня листьев. Выделяют два способа реализации иерархической кластеризации: алгомеративный (при котором индивидуальные образцы данных формируют кластеры, объединяющиеся далее во всё более крупные группы) и разделительный (при котором сначала вся популяция разделяется на крупные группы, а затем рекурсивно каждая из таких групп разделяется на всё более мелкие).

Центроидные алгоритмы определяют кластеры по расстоянию от так называемых центроидов — геометрических центров кластера, зависящих от выбранной метрики расстояния. Несмотря на простоту имплементации центроидного подхода и преимущество в скорости расчёта по сравнению с остальными методами кластеризации, у этого вида алгоритмов есть существенный недостаток: для центроидных алгоритмов необходимо заблаговременно указывать количество кластеров в системе, число которых далеко не всегда бывает известно заранее.

В отличие от предыдущих двух подходов, метод кластеризации по распределению позволяет находить группы различной формы (в соответствующем векторном пространстве) в зависимости от заранее заданного распределения. Однако необходимость предварительно задавать форму распределения при этом является весомым недостатком метода, сужающим область его применения.

Наконец, кластеризация по плотности позволяет выделить группы разнообразной формы, которую не нужно указывать для работы алгоритма. Кроме того, количество кластеров также определяется во время выполнения алгоритма. Обозначенные преимущества этого подхода даются ценой больших вычислительных затрат, из-за которых алгоритмы кластеризации по плотности являются самыми медленными из описанных методов кластеризации.

Динамика агентов под воздействием некоторых управляющих алгоритмов может приводить к изменению количества кластеров с течением времени (кластерным потокам). Таким образом, иерархический подход лучше всего подходит для поиска кластеров в MAC, поскольку позволяет автоматически определять количество кластеров. Альтернативным выбором послужит метод кластеризации по плотности, также способный работать без предварительного задания количества кластеров. Однако группы агентов во фреймворке кластерных потоков определяются лишь близостью состояний агентов (но не формой кластеров), поэтому не возникает необходимость использовать вычислительно трудные методы кластеризации по плотности.

2. Фреймворк кластерных потоков

Для лучшего понимания работы алгоритма кластерного управления необходимо проанализировать поведение соответствующей управляемой мультиагентной системы. Описываемый далее фреймворк позволяет выявить особенности динамики осцилляторов Курамото: в частности, определить, при каких параметрах кластерная синхронизация (в терминах фреймворка) будет сохраняться, а при каких — нарушаться. Нарушение кластерной синхронизации означает изменение кластерной структуры: агенты из одного кластера могут присоединиться к другим, что усложняет задачу кластерного управления в общем виде, поскольку количество и состав кластеров может меняться во времени.

2.1. Модель MAC с кластеризацией

Следуя идеям, описанным в [46], всюду далее рассматриваются неизолированные системы, состоящие из агентов, эволюция которых определяется их текущим состоянием, состоянием других агентов в системе, а также внешним воздействием окружающей среды. Модель (2) удобно модифицировать следующим образом:

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t), U_i(t), \eta_i(t)), \tag{5}$$

где $x_i(t) \in \mathbb{R}^{n_i}$ является вектором состояния агента $i \in \mathcal{N}$; $u_i(t)$ — локальное управление, описывающее, как взаимодействия между агентами влияют на их состояние; $U_i(t)$ является кластерным управлением, влияющим на большие группы агентов единовременно; $\eta_i(t) \in \mathbb{R}^{m_i}$ — стохастическая переменная.

Сеть агентов моделируется ориентированным графом взаимодействий: $\mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t))$, где $\mathcal{E}(t)$ — совокупность его направленных дуг, а $\mathcal{N}$ обозначает набор вершин-агентов. Обозначим множество соседей (окрестность) агента i как $\mathcal{N}_i(t) \subseteq \mathcal{N}$. Это означает, что каждый агент j из $\mathcal{N}_i(t)$ может взаимодействовать с i в момент времени t.

Степень захода вершины i ($i \in \mathcal{E}$) определяется как $d(i) = \sum_{l=1}^{N} A_{il}$,

где A — матрица связности $\mathcal{G}$. Другими словами, степень захода i это взвешенная сумма числа агентов в $\mathcal{N}_i$. Аналогично определяется степень захода вершины i за вычетом вершины j $(i, j \in \mathcal{E})$ как $d_j(i) = \sum_{l=1, l\neq j}^{N} A_{il}$. Полезно определить сильно-связный граф: это такой граф, где для некоторого n существует "путь'' между любыми двумя вершинами.

Определение 1. *Функция $g_i(x_i(t), \nu_i(t))$ называется выходом агента i если $g : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \mapsto \mathbb{R}^l$, где l не зависит от i.*

Как обсуждалось ранее, выходы агентов необходимо задавать для наблюдения за их состоянием. Пусть $y_j(t) = g_j(x_j(t), \nu_j(t))$ $j \in \mathcal{N}_i(t)$ — выход агента j из окрестности i, используемый для коммуникаций при локальных взаимодействиях. Под "связью" между агентами понимается, что решение i (в момент времени t) основано на выходах $y_j(t)$. На практике эти выходы могут передаваться от j к i или же показываться агентом j, будучи затем распознанными агентом i. Математически правила коммуникации строго задаются в u_i (см. (5)):

$$u_i(t) = f_i\left(\{y_j(t)\}_{j\in\mathcal{N}_i(t)}\right), \tag{6}$$

где $f_i(\cdot)$ — функция от выходов $y_j(t)$ $j \in \mathcal{N}_i(t)$. Уравнение (6) называют *протоколом связи* в том смысле, что оно содержит правила управления i на основе выходов j, полученных агентом i. Таким образом, вводится определение мультиагентной сети:

Определение 2. *Тройка объектов, состоящая из: 1) семейства агентов* (5)*; 2) графа взаимодействий $\mathcal{G}$ и 3) протокола связи, определенного в* (6)*, называется мультиагентной сетью.*

В дальнейшем многоагентная сеть будет обозначаться буквой $\mathcal{N}$, соответствующей множеству агентов сети.

Пусть $z_i(t) = h_i(x_i(t), \nu_i(t))$ — выход i, используемый для измерения синхронизации.

Определение 3. *Пусть* $\Delta_{ij}(t_*) = \|z_i(t_*) - z_j(t_*)\|$ *— разность между значениями выходов* z_i *и* z_j *в момент времени* t_**, где* $\|\cdot\|$ *— соответствующая норма. Тогда:*

1. *агенты* i *и* j *являются синхронизированными, иначе говоря, достигают консенсуса в момент времени* t_**, если* $\Delta_{ij}(t_*) = 0$*; аналогично, агенты* i *и* j *являются асимптотически синхронизированными, если* $\Delta_{ij}(\infty) = \overline{\lim}_{t_*\to\infty}\Delta_{ij}(t_*) = 0$*;*

2. *агенты* i *и* j *являются* ε*-синхронизированными, иначе говоря, достигают* ε*-консенсуса в момент времени* t_**, если* $\Delta_{ij}(t_*) \leq \varepsilon$*; аналогично, агенты* i *и* j *являются асимптотически* ε*-синхронизированными если* $\Delta_{ij}(\infty) = \overline{\lim}_{t_*\to\infty}\Delta_{ij}(t_*) \leq \varepsilon$*;*

Как было описано во введении, агенты могут синхронизироваться не только глобально, но и покластерно.

Определение 4. *Семейство подмножеств* $\mathcal{M}(t) = \left\{\mathcal{M}_\alpha(t) : \mathcal{M}_\alpha(t) \subseteq \mathcal{N}\ \forall t \geq 0\ \forall \alpha \in \overline{1, m(t)}\right\}_{\alpha=1}^{m(t)}$ *множества* $\mathcal{N}$ *является разбиением над* $\mathcal{N}$ *если выполнены нижеследующие условия.*

1. *Все подмножества* $\mathcal{M}_\alpha(t)$ *непустые.*

2. $\bigcup_{\alpha=1}^{m(t)} \mathcal{M}_\alpha(t) = \mathcal{N}\ \forall t \geq 0.$

3. $\mathcal{M}_\alpha(t) \cap \mathcal{M}_\beta(t) = \varnothing\ \alpha \neq \beta.$

Таким образом, предлагается дополнительное определение для конкретного случая кластерной синхронизации.

Определение 5. *Мультиагентная сеть с разбиением* $\mathcal{M}(t_*)$ *над* $\mathcal{N}$ *является* (ε, δ)*-синхронизированной, иначе говоря, достигает* (ε, δ)*-консенсуса в момент времени* t_* *для некоторых* $\delta \geq \varepsilon \geq 0$ *если*

1. $\Delta_{ij}(t_*) \leq \varepsilon$ *для агентов* i *и* j *из одного кластера.*

2. $\Delta_{ij}(t_*) > \delta$ *для агентов* i *и* j *из разных кластеров.*

Случай $(0, 0)$*-синхронизации, таким образом, называется кластерной синхронизацией. Говорят, что* $\mathcal{M}(t_*)$ — кластеризация *над* $\mathcal{N}$*.*

2.2. Анализ модели Курамото

Перед осуществлением наблюдений за пространством состояний системы осцилляторов Курамото её следует представить в каноническом виде в соответствии с (5) и определением мультиагентной сети. Анализ модели средствами фреймворка кластерных потоков позволит обнаружить особенности её поведения, что полезно при тестирования алгоритма кластерного управления.

Граф связей между агентами-осцилляторами $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ — пусть его топология для простоты не зависит от времени. Соответствующая ему матрица связности обозначается Υ, элемент Υ_{ij} которой принимает значения из множества $\{0, 1\}$ для любой пары агентов i и j. Значение 0 может пониматься как "агент j не может связаться с агентом i ($j \notin \mathcal{N}_i$)", а 1 означает "сигналы от агента j доходят до i ($j \in \mathcal{N}_i$)". Поскольку локальное управление (протокол) зависит от выходов агентов (6), следует обозначить выходы: $y_i(t) = \theta_i(t)$.

Пусть кластеризация $\mathcal{M}(t_1)$ возникла в системе осцилляторов в момент времени t_1 и остается постоянной на временном интервале $T = [t_1, +\infty)$, а $t \in T$. Введём следующие функции локального и мезоскопического управления:

$$\begin{gathered} u_i(t) = w_i + \rho \sum_{j=1}^{N} \Upsilon_{ij} \sin\left(\theta_j(t) - \theta_i(t)\right), \\ U_i(t) = \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha), \end{gathered} \tag{7}$$

где $\rho > 0$ — константа, w_i — собственная частота осциллятора, $\mathcal{F}_\alpha(\cdot)$ — мезоскопическая функция одинаковая для всего кластера $\mathcal{M}_\alpha$, μ_i — чувствительность агента к управляющей функции $\mathcal{F}_\alpha(\cdot)$. Кроме времени t в аргументы $\mathcal{F}_\alpha(\cdot)$ может входить $\overline{x}_\alpha(t)$, содержащий интегральные характеристики кластера α: примером такой характеристики может служить положение центроида кластера. Следует отметить, что в системах с большим числом агентов в кластере его интегральные характеристики слабо зависят от состояния индивидуальных агентов внутри него.

Пусть оба управляющих действия входят в новую модель аддитивно

($i \in \mathcal{M}_\alpha$). Таким образом модель Курамото в каноническом мультиагентном представлении с добавлением функции кластерного управления:

$$\dot{\theta}_i(t) = \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha(t)) + w_i + \rho \sum_{j=1}^{N} \Upsilon_{ij} \sin\left(\theta_j(t) - \theta_i(t)\right), \tag{8}$$

где $\rho\Upsilon_{ij}$ имеет вид K_{ij} в классической модели (4).

В действительности функции управления (7) сохраняют кластерную структуру только если соблюдены некоторые условия. Во-первых, синхронизация зависит от значений μ_i: так, некоторые агенты в кластере $\mathcal{M}_\alpha$ могут реагировать на управление $\mathcal{F}(\cdot)$ достаточно сильно, чтобы "сломать'' структуру кластера. Во-вторых, большая разница значений w_i также может привести к нежелательным последствиям вплоть до хаотического поведения системы. Сформулируем теорему об ограничениях на параметры модели (8), обобщающую описанные выше случаи.

Пусть в системе установилось состояние (ε, δ)-синхронизации. Динамика некоторых агентов в кластере $\mathcal{M}_\alpha$ под управлением $\mathcal{F}(\cdot)$, однако, может оказаться деструктивной для всего кластера в случае, если значения их чувствительностей будут сильно отличаться от чувствительности других агентов в кластере. Кроме того, значительная разница в значениях w_i также может привести к нежелательным последствиям, вплоть до хаотического поведения системы. Нижеследующая теорема формализует описанные сценарии для значений $\varepsilon = \delta = 0$, приводя условия на параметры модели, достаточные для сохранения $(0, 0)$-синхронизации.

Теорема 1. *Рассмотрим мультиагентную сеть* (8). *Пусть* $t \in T$, *выход* $z_i(t) = \dot{\theta}_i(t)$ *и* $\Delta_{ij}(t) = |z_i(t) - z_j(t)|$. *Пусть также* $\mathcal{F}_\alpha$ *не зависит от* θ_i $\forall i$. *Нижеследующие условия достаточны для сохранения* $(0, 0)$-*синхронизации.*

1. *В случае* $i, j \in \mathcal{M}_\alpha$

$$|w_i - w_j + (\mu_i - \mu_j)\mathcal{F}_\alpha| \leq \rho C_{ij} \sum_{l=1}^{N} [\Upsilon_{il} + \Upsilon_{jl}], \tag{9}$$

где $C_{ij} = 1$ *при* $\Upsilon_{ij} = \Upsilon_{ji} = 0$*; иначе*

$$C_{ij} = \max\left\{\sqrt{1-(\Gamma_i(j))^2}, \sqrt{1-(\Gamma_j(i))^2}, \frac{\sqrt{2}}{2}\right\}, \tag{10}$$

где

$$\Gamma_i(j) = \frac{-d_i(j) + \sqrt{(d_i(j))^2 + 8(\Upsilon_{ij} + \Upsilon_{ji})^2}}{4(\Upsilon_{ij} + \Upsilon_{ji})}. \tag{11}$$

2. *Для* $i \in \mathcal{M}_\alpha$, $j \in \mathcal{M}_\beta$, $\alpha \neq \beta$

$$|w_i - w_j + \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha) - \mu_j \mathcal{F}_\beta(t, \overline{x}_\beta)| > 0. \tag{12}$$

3. *Граф* $\mathcal{G}$ *сильно связный.*

Доказательство теоремы приведено в Приложении.

3. Алгоритм кластерного управления

С целью смоделировать удалённые наблюдения за мультиагентной системой, динамические траектории “квантуются”: континуальное исходное пространство состояний переводится в дискретное счётное пространство. Мощность нового дискретного пространства зависит от разрешающей способности и “поля зрения” сенсора-наблюдателя, определяющей шаг дискретизации. В контексте данной работы для проведения симуляций параметры разрешающей способности и поля зрения выбираются эвристически, основываясь на конкретном поведении системы. Для лучшего понимания концепта наблюдателя можно рассмотреть в качестве примера цифровой фотоаппарат или видеокамеру: подобно съёмке сцены сенсор-наблюдатель регистрирует определённую область пространства (состояний) с некоторой степенью дискретизации. На текущий момент этап ограничения и квантовании пространства состояний выполняется эвристически, что порождает задачу оптимального автоматического выбора параметров масштабирования области наблюдений и выбора шага дискретизации.

Алгоритм состоит из трёх основных шагов:

1) ограничить и квантовать пространство состояний (смоделировать удалённый сенсор);

2) провести наблюдения со сжатием за вектором состояний;

3) найти кластеры;

4) в случае необходимости изменить стратегию кластерного управления.

Обозначенные шаги циклически повторяются, что позволяет обрабатывать потоковые данные.

3.1. Ограничение и квантования динамических траекторий

Рассмотрим простейший пример одномерных синхронизационных выходов $\{z_i(t)\}_{i\in\mathcal{N}}$. В этом случае траектории можно изобразить кривыми на "координатной плоскости", горизонтальная ось которой является временем t, а вертикальная — z. Полуплоскость $t > 0$ называется *полным пространством состояний* и обозначаем её $\mathfrak{S}$. С точки зрения удалённого сенсора полуплоскость $\mathfrak{S}$ является набором изменяющихся во времени сцен (одномерных реализаций состояний агентов). Однако, поскольку выходы $z_i(t)$ в общем случае могут принимать любые значения, практически объять сенсором всю бесконечно простирающуюся сцену невозможно, в связи с чем следует выбрать ту её область, где располагаются траектории в конкретной задаче. Это равносильно установке "поля зрения" сенсора — ограничения по z значениями $Z_{\min}$ и $Z_{\max}$.

Кроме того, для реализации прототипа в текущей работе следует задать начало наблюдений и их конец (ограничения по t) — $t_{\min}$ и $t_{\max}$. Таким образом, для любого момента времени $t \in [t_{\min}, t_{\max})$ точка траектории $z_i(t)$ принадлежит $\big[Z_{\min}, Z_{\max}\big)$. Данное ограничение по времени не является обязательным при реализации потокового варианта алгоритма.

Благодаря описанным выше действиям в полуплоскости $\mathfrak{S}$ удаётся выделить ограниченную область: $\mathfrak{R} = \big[Z_{\min}, Z_{\max}\big) \times [t_{\min}, t_{\max}) \subset \mathfrak{S}$. Значения $Z_{\max}$ и $t_{\max}$ не принадлежат $\mathfrak{R}$ с целью упрощения дальнейших конструкций. Дальнейшая дискретизация $\mathfrak{R}$ предполагает разбиение этой области на "клетки" (семплирование):

$$\begin{aligned} \big[Z_{\min}, Z_{\max}\big) &= \big[Z_{\min}, Z_1\big) \cup \ldots \cup \big[Z_{p-1}, Z_{\max}\big), \\ [t_{\min}, t_{\max}) &= [t_{\min}, t_1) \cup \ldots \cup [t_{q-1}, t_{\max}), \end{aligned} \tag{13}$$

где значения p и q определяются шагами дискретизации: пространственной разрешающей способностью сенсора $Z_r = Z_i - Z_{i-1}$ и длины выдержки $t_e = t_j - t_{j-1}$. Значения Z_r и t_e зависят от формы траекторий

$\{z_i(t)\}_{i\in\mathcal{N}}$ и выбираются эмпирически. Например, интервал времени t_e должен быть сильно короче длительности интересующего динамического события, чтобы удалось запечатлеть быстрые изменения состояний. Такие события могут включать внутрикластерные возмущения или межкластерный обмен агентами.

Таким образом, для реализации наблюдений за траекториями моделируется p-пиксельный однобитный сенсор, принимающий значение 1 в месте регистрации траектории, иначе 0. Серию из q наблюдений удобно записать в виде матрицы $\mathfrak{B} \in \{0,1\}^{p\times q}$, столбцы которой можно подвергать сжатию в соответствии с (3). Ясно, что при выборе меньших значений Z_r и t_e размерность $\mathfrak{B}$ возрастает, что приводит к увеличению разрешения и читабельности соответствующих портретов дискретных траекторий, однако при этом увеличивают время сжатия и реконструкции наблюдений.

Для последующего поиска кластеров реконструированная матрица $\widehat{\mathfrak{B}}$ пороговой фильтрации с адаптивной регулировкой порога: её элементы $\hat{b}_{i,j}$ приравниваются к единице, если они превышают некоторое пороговое значение, иначе приравниваются к 0. В качестве этого значения можно выбрать пропорциональное среднему значению всех элементов в столбце $5 \cdot \hat{b}_{\cdot,j}$ — как будет показано далее в симуляциях, это пороговое значение является хорошей эвристикой.

3.2. Опознание со сжатием

В процессе симуляций заполняется записывающая матрица $\mathfrak{B}$. Каждый из её столбцов, моделирующий "снимок" сенсора, подвергается компрессии при помощи умножения справа на постоянную матрицу измерений: $y = Ax$, где y — вектор сжатых наблюдений. Предварительно генерируется матрица A, элементы которой берутся из нормального распределения

$$a_{ij} = \mathcal{N}\left(0, \frac{1}{m}\right)$$

в соответствии с [24]. Также предварительно, исходя из усреднённой разреженности столбцов $\mathfrak{B}$, определяется величина $m = c \cdot s_0$, где c

принимает значения от 2 до 4, а s_0 — средняя (по всем столбцам) разреженность столбцов $\mathfrak{B}$. По величине m и размерности вектора x, равной p (количеству пикселей сенсора), можно определить степень сжатия: $\gamma_c = p/m$, где c — упомянутая выше константа.

3.3. Поиск кластеров

Задача поиска кластеров на текущий момент сводится к задаче определения количества найденных кластеров и расположения их центроидов. Исходные данные представляют из себя бинарные разреженные векторы, что позволяет ввести евклидову метрику, которая по двум ненулевым элементам такого вектора выдаёт модуль разности их индексов. Центроид кластера в этом случае представляют из себя среднее значение индексов, попавших в кластер ввиду их близости по описанной выше метрике.

Иерархическая кластеризация позволяет выделять кластеры безотносительно к действительному их количеству, в связи с чем предсказанное количество кластеров может не совпадать с истинным значением. Используя этот факт, можно построить метрику для оценки точности кластеризации. Введём следующую переменную:

$$\chi[i] = \begin{cases} 1, & \text{если } \widehat{M}[i] = M, \\ 0, & \text{если } \widehat{M}[i] \neq M, \end{cases}$$

где i — номер наблюдения из серии (принимает значения от 1 до q), $\widehat{M}$ — предсказанное алгоритмом оценочное количество кластеров, M — истинное количество кластеров. В таком случае процентная точность по q наблюдениям

$$\epsilon = \frac{\sum_{i=1}^{q} \chi[i]}{q} \cdot 100\%. \tag{14}$$

В некоторых случаях ℓ_1-реконструкция сжатых векторов состояний агентов может привести к появлению нежелательных всплесков в местах, где отсутствуют траектории. Эти обособленные выбросы, как правило, распознаются как малые кластеры, состоящие из 1–2 элементов.

Как будет показано далее, точность алгоритма отыскивания кластеров может быть увеличена при помощи устранения таких выбросов из общего списка найденных кластеров. Всюду далее величина минимального кластера обозначается $M_{\min}$.

3.4. Оценка сложности

Большая размерность пространства состояний в мультиагентных системах приводит возникновению в них непредсказуемого эмерджентного поведения. Это означает, что передача данных из микро-уровня на макро-уровень может являться своеобразным "бутылочным горлышком" во всём процессе коммуникаций между системой и наблюдателем (дата-центром). Из-за этого явления система оказывается сложной с точки зрения запутанности связей между её составляющими, что приводит к проблеме анализа больших данных, обмен которыми происходит в МАС. Таким образом, сложность предлагаемого алгоритма корректнее всего оценить в количестве собранных данных. Рассмотрим столбец $b_{\cdot j}$ матрицы $\mathfrak{B}$, являющийся вектором в p-мерном пространстве. Опознание со сжатием позволяет уменьшить объём передаваемых данных с кратного p до кратного $s\log(p/s)$, где s равняется количеству ненулевых компонент разреженного сигнала, представляемого вектором $b_{\cdot j}$. В случае больших пространств состояний опознание со сжатием позволяет расширить упомянутое "бутылочное горлышко" пропорционально разреженности, чтобы затем быстрее синтезировать кластерное управление.

4. Требования к ПО

В этом разделе сформулированы требования к прототипу программного обеспечения, реализующего описанный выше алгоритм. Требования получены исходя из анализа недостатков методов управления мультиагентными системами, обозначенных в Главе 1. Также требования основываются на описанном Главе 3 алгоритме управления.

4.1. Функциональные требования

4.1.1. Динамика мультиагентной системы

Для симуляции динамики МАС в первую очередь следует обеспечить возможность задать количество агентов в системе $0 \leq N \leq 100$, а также граф связей между ними, например, в виде матрицы связности или ассоциативного массива. Поскольку вместо симулятора в будущем можно будет применять алгоритм к реальным киберфизическим системам, то симуляции следует проводить при отсчёте времени в секундах — для этого нужно обеспечить возможность задавать элементарный шаг по времени dt, используемый при численном решении систем дифференциальных уравнений, описывающих динамику системы агентов.

С целью упростить тестирование начальные состояния агентов должны задаваться не только генератором псевдослучайных чисел, но и вручную — например, для проверки особых случаев неустойчивого равновесия в системе. Следует также обеспечить возможность точно задавать чувствительность агентов к локальному и кластерному управляющим входам. Кроме того, работа системы не должна зависеть от природы агента — в связи с этим должна предоставиться возможность имплементировать агентов с желаемым поведением.

Для самих управляющих алгоритмов следует обеспечить возможность их простой имплементации (и изменения) программными средствами разрабатываемой системы с указанием всех необходимых параметров вручную.

4.1.2. Удалённые наблюдения со сжатием

Удалённые наблюдения за MAC сходны с процессом фотосъёмки классическим цифровым фотоаппаратом. В связи с этим следует обеспечить возможность задавать следующие параметры: количество "пикселей" в сенсоре, время экспозиции, а также обобщённую кратность приближения, выражаемую в задаваемых границах наблюдения (с целью упрощения симуляции оптики).

Для сжатия в соответствии с методологией Compressive Sensing нужно вне сенсора задать матрицу наблюдений (которую затем следует также предоставить в дата-цетр для реконструкции) и, соответственно, размерность вектора сжатых наблюдений. Для кэширования и мониторинга нужно обеспечить возможность записывать "снимки" — оригинальные и сжатые данные из сенсора.

4.1.3. Реконструкция данных о системе

Поскольку реконструкция должна осуществляться в удалённом дата-центре, то его имплементацию следует отделить от имплементации сенсора, позволив им общаться между собой при помощи некоторого канала коммуникаций. На текущий момент не ставится задача обеспечить между ними связь через сеть, поэтому достаточно передавать записанные данные из сенсора в дата-центр локально.

Для реконструкции следует обеспечить возможность предоставить дата-центру ту же матрицу наблюдений, что предоставлялась сенсору.

4.1.4. Поиск кластеров

Поиск кластеров предполагает предварительное задание размера минимального кластера (для отсеивания выбросов) и пороговое значение расстояния между кластеризуемыми элементами. Если расстояние между двумя элементами превышает это пороговое значение, то их следует отнести к разным кластерам. Таким образом, следует обеспечить возможность выбирать размер минимального кластера и порогового значения.

Кроме того, помимо определения принадлежности тех или иных элементов к некоторому кластеру, алгоритм должен определять положение центроида кластера, основываясь на координатах входящих в него элементов.

4.1.5. Формирование стратегии кластерного управления

Обеспечение возможности формирования кластерного управления на основе конфигурации распознанных кластеров сводится к требованию о возможности изменения соответствующей стратегии программными средствами разрабатываемого ПО. Таким образом, программное обеспечение должно быть способно решать заданную пользователем задачу управления.

4.2. Нефункциональные требования

Программное обеспечение предназначается для установки и использования на персональных компьютерах пользователя, а также в будущем для развёртывания на компактных устройствах с низким потреблением энергии. В связи с этим выдвинуты следующие требования:

- ПО должно быть кросс-платформенным (в частности, работать на компьютерах под управлением ОС на базе ядра Linux или Windows 7 и выше);
- при работе ПО требует не более 500 мегабайт оперативной памяти;
- ПО со всеми используемыми библиотеками требует не более 200 мегабайт постоянной памяти.

5. Особенности реализации

Поскольку задачей текущей работы является разработка *прототипа* программного обеспечения, реализующего описанный выше алгоритм, то должны быть учтены следующие требования к среде разработки и языку программирования

1. Простота написания и чтения кода.
2. Удобство отладки и сопровождения ПО.
3. Возможность простой визуализации результатов вычислений.

Для разработки ПО, удовлетворяющего перечисленным выше требованиям, лучше всего подходят языки MATLAB и Python. Каждый из них имеет свои преимущества (в сравнении друг с другом).

Преимущества MATLAB:

- производительность;
- специализированные библиотеки для задач оптимизации;
- простое локальное развёртывание среды разработки;
- простая переносимость кода.

Преимущества Python:

- свободное ПО с открытым исходным кодом;
- развёртывание в облаке;
- ООП;
- низкое потребление постоянной и оперативной памяти.

Под развёртыванием в облаке имеется в виду в первую очередь PaaS Google Colab, предоставляющий ресурсы для запуска кода Python в интерактивной среде Jupyter Notebook. Стоит отметить, что разработчики MATLAB тоже предоставляют PaaS для разработки в облаке, однако, поскольку MATLAB является проприетарным ПО, доступ к этой услуге возможен только при наличии соответствующей лицензии. Несмотря на наличие в MATLAB множества специализированных библиотек для решения задач оптимизации, такие библиотеки для Python

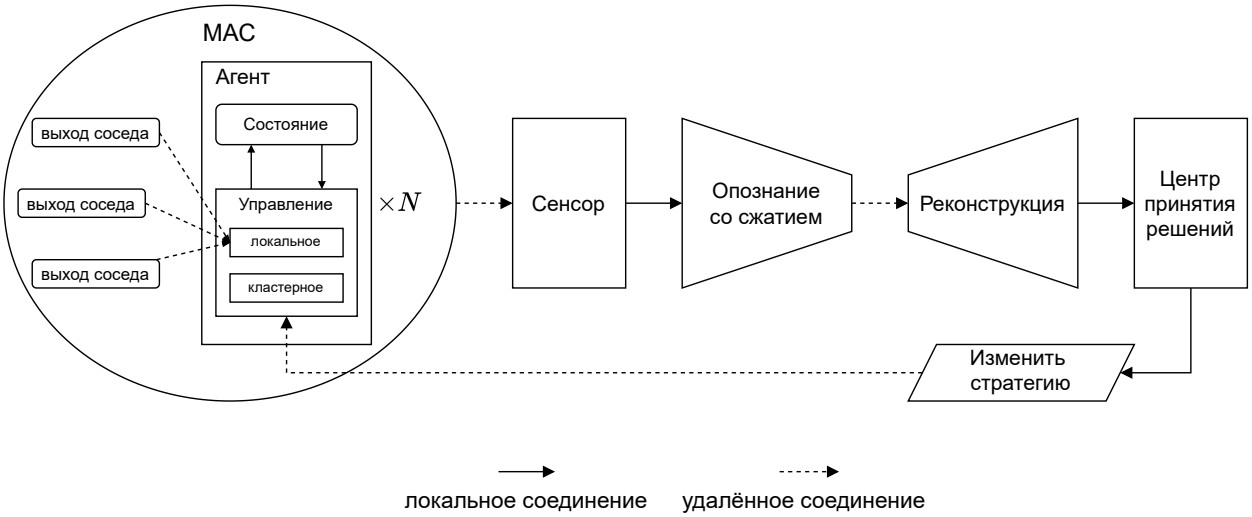

Рис. 1: Архитектура ПО для кластерного управления MAC

как SciPy и NumPy способны воспроизвести весь необходимый функционал для реализации алгоритма кластерного управления. Более того, среда для разработки MATLAB, без которой невозможно исполнение кода на этом языке, требует больших ресурсов постоянной памяти, из-за чего использование этого языка для разработки под системы с низким потреблением энергии является нецелесообразным.

На Рисунке 1 представлена архитектура ПО, реализующая алгоритм кластерного управления. Соответствующее разработанное ПО находится в репозитории [56] — программное обеспечение готово к загрузке в качестве библиотеки и сопровождено примером использования. В репозитории также содержится ссылка для онлайн-запуска симуляций в облаке Google Colab. Версия для Google Colab имеет упрощённый исполняемый код ПО, реализующего достаточный для демонстрации алгоритма функционал, однако при этом ценой утраты масштабируемости.

5.1. Симуляция динамики агентов

Целью проведения симуляций является численное моделирование решения системы уравнений, описывающих динамику агентов (5):

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t), U_i(t), \eta_i(t)).$$

Для этого следует предварительно задать не только параметры самой динамической системы, но и параметры симулятора: дискретный шаг по времени, длительность симуляции и соответствующее количество итераций. Таким образом, получается система разностных уравнений:

$$x_i[k+1] = f_i(x_i[k], u_i[k], U_i[k], \eta_i[k]) \cdot dt, \tag{15}$$

где k — номер текущей итерации, dt — конечный шаг по времени.

Для реализации полученной системы подходит библиотека NumPy, позволяющая осуществлять матрично-векторные вычисления с высокой производительностью. NumPy использует возможности BLAS (Basic Linear Algebra Subprograms), оптимизирующих матрично-векторные операции под аппаратное обеспечение с помощью распараллеливания и кэширования.

Симуляция производится при помощи трёх классов: “симулятора”, “агента” и “контроллера”. Симулятор инициализирует необходимое количество агентов, хранит их идентификатор и граф связей. Каждый агент инициализирует два контроллера — для локального и кластерного управления, при этом повлиять на состояние агента можно только при помощи этого контроллера. Сами агенты не хранят в себе информацию о своём идентификаторе и графе связей — в будущем это поможет позволить перейти к полной децентрализации системы.

5.2. Визуализация данных

Полученные в ходе симуляций данные можно визуализировать в Jupyter Notebook при помощи таких библиотек, как matplotlib, Seaborn и Plotly. При этом Seaborn и Plotly предлагают избыточный функционал (например, интерактивные графики), в связи с чем для визуализации выбран matplotlib. Визуализация используется в демонстрационном примере работы системы в репозитории.

5.3. Реализация опознавания по сжатию

Для восстановления $\hat{x}$ используется решающий алгоритм на основе метода внутренней точки, реализованный библиотекой SciPy. Реконструкцию $\hat{x}$ также можно реализовать при помощи библиотек для вычислений на GPU: TensorFlow или PyTorch, добивающихся таким образом наилучшей производительности векторно-матричных вычислений. Однако синтаксис SciPy намного проще, а сама библиотека занимает значительно меньше постоянной памяти, чем перечисленные выше аналоги для расчётов на GPU, поэтому в угоду читабельности кода и простоты эксплуатации прототипа ПО выбор остановился на нём. При этом время расчёта оценки $\hat{x}$ с использованием SciPy составляет порядка секунд (на 12-ядерном процессоре с частотой 3.6 ГГц), в связи с чем нет необходимости ускорения на GPU. Стоит также отметить, что в системах с низким потреблением энергии (например, в компьютерах Raspberry Pi) может вовсе отсутствовать графический процессор, что сделает использование TensorFlow или PyTorch нецелесообразным.

Имплементация реконструирующего дата-центра осуществлена отдельно от имплементации сенсора, что может позволить в будущем физически разделить устройства-сенсоры от дата-центров и передавать потоковые данные на расстоянии.

5.4. Имплементация алгоритма кластеризации

Библиотека SciPy позволяет реализовать иерархическую кластеризацию. Также с этой задачей можно справиться при помощи TensorFlow и PyTorch, однако, поскольку поиск кластеров выполняется за время порядка секунд, то отсутствует нужда использовать ускорение на GPU. С помощью описанной выше библиотеки matplotlib производится визуализация центроидов кластеров вместе с визуализацией матрицы $\mathfrak{B}$.

6. Апробация

Для апробации разработанного и реализованного алгоритма применяется модифицированная модель Курамото с кластерным управлением. На данном этапе главной целью апробации является проверка алгоритма поиска кластеров по удалённым наблюдениям, а также определение зависимости точности кластеризации от степени сжатия. В связи с этим кластерное управляющее воздействие определяется заранее заданным истинным количеством кластеров, которое требуется предсказать с помощью иерархической кластеризации. В будущем кластерное управляющее воздействие можно будет синтезировать уже не по заранее определённым кластерам, а по предсказанным — на данный момент это находится за рамками текущей работы.

6.1. Симуляция модели осцилляторов Курамото

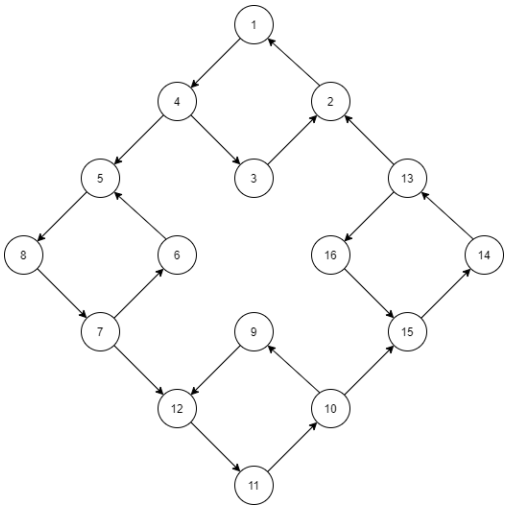

Рис. 2: Топология используемого графа

Рассмотрим модель (8) и ее симуляцию на интервале $T = [0, 60]$:

$$\dot{\theta}_i(t) = \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha(t)) + w_i + \rho \sum_{j=1}^{N} \Upsilon_{ij} \sin\left(\theta_j(t) - \theta_i(t)\right).$$

Возьмём синусоидальный алгоритм кластерного управления:

$$U_i = \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha(t)) = \mu_i \sin(2\pi f_\alpha(t - 20)) \tag{16}$$

в качестве примера, где $U_i(t)$ "включается" с момента времени $t = 20$.

Зададим следующие параметры модели:

- $N = 16$;
- топология графа $\mathcal{G}$ как на Рисунке 2;
- начальные фазы $\theta_i(0)$ из равномерного распределения на окружности S^1;
- значения f_α берутся из равномерного распределения на $[0, 1]$;
- $\rho = 0.5$;
- собственные частоты w_i представлены в Таблице 1;
- значения μ_i представлены в Таблице 2.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
4.1	4.2	4.3	4.4	8.1	8.2	8.3	8.4
w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
12.1	12.2	12.3	12.4	16.1	16.2	16.3	16.4

Таблица 1: Значения w_i, используемые в симуляции

μ_1	μ_5	μ_9	μ_{13}	μ_2	μ_6	μ_{10}	μ_{14}
0.375				0.75			
μ_3	μ_7	μ_{11}	μ_{15}	μ_4	μ_8	μ_{12}	μ_{16}
1.125				1.5			

Таблица 2: Значения μ_i, используемые в симуляции

Далее кластерную синхронизацию агентов-осцилляторов в модели Курамото будем определять по близости значений их угловых частот $\dot{\theta}_i$,

играющих роль соответствующих выходов z_i в теореме из Главы 2. Симуляция динамики 16 агентов позволяет получить траектории изменения их угловых частот (см. Рисунок 3). Из Теоремы 1 (уравнений (9) и (12)) следует, что представленные значения параметров позволяют кластерной картине оставаться неизменной с течением времени. На Рисунке 3 траектории агентов из одного кластера помечены одинаковым цветом для наглядности.

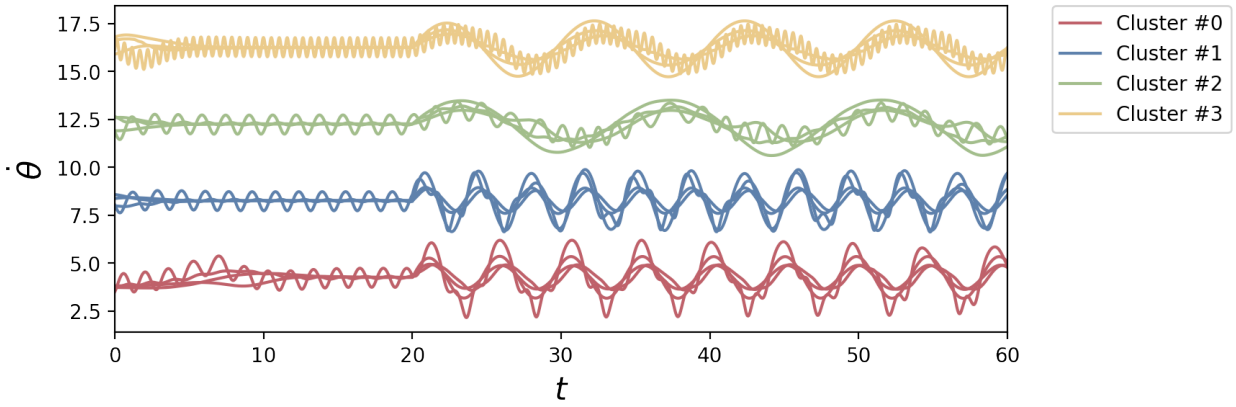

Рис. 3: Траектории угловых частот 16 агентов-осцилляторов, формирующих 4 кластера, при синусоидальном кластерном управлении, описанном в уравнении (16)

Несмотря на достаточно небольшое количество агентов, дискретное пространство состояний такой системы тем не менее оказывается довольно большой размерности. Если рассмотреть визуализацию этого пространства состояний далее, то станет ясно, что такое пространство может включать и сильно большее количество агентов. Однако после кластерной синхронизации агенты в любом случае займут относительно малую зону пространства (таким образом приводя пространство состояний к разреженному виду). Таким образом, на текущий момент используется простой пример с 16 агентами исключительно для демонстрации алгоритма, при этом система остаётся хорошо масштабируемой.

6.2. Определение кластеров по дискретным наблюдениям

Обозначим эмпирически выбранные параметры системы наблюдений из сенсора и записывающего устройства (параметры дискретизации пространства состояний):

- $Z_{\min} = 0$, $Z_{\max} = 20$, $Z_r = 0.1$;
- $t_{\min} = 20$, $t_{\max} = 60$, $t_e = 0.1$.

При этих параметрах производится симуляция 200-пиксельного сенсора, с помощью которого совершается 400 одномерных "снимков" динамических траекторий (угловых частот) за промежуток времени от 20 до 60 секунд от начала симуляции системы осцилляторов. Эти снимки можно представить в виде матрицы $\mathfrak{B}$ размерности 200×400, построенной согласно алгоритму в Главе 3. На Рисунке 4 изображена полученная в симуляции матрица $\mathfrak{B}$, где белые пиксели отвечают за наличие одного или нескольких агентов в соответствующей области пространства состояний.

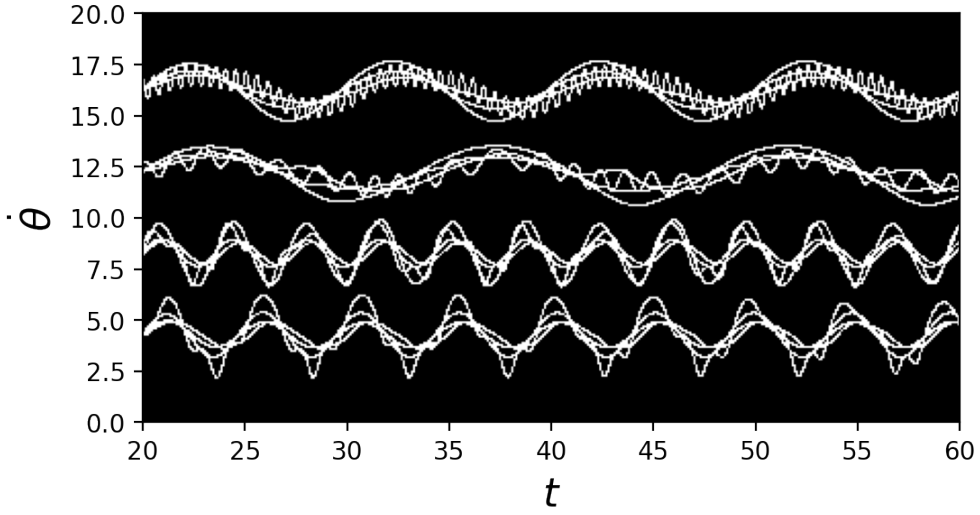

Рис. 4: 400 одномерных последовательных 200-пиксельных снимков траекторий 16 осцилляторов, представленных в виде столбцов матрицы $\mathfrak{B}$ (см. алгоритм в Главе 3); белые пиксели отвечают за наличие одного или нескольких агентов с соответствующим состоянием

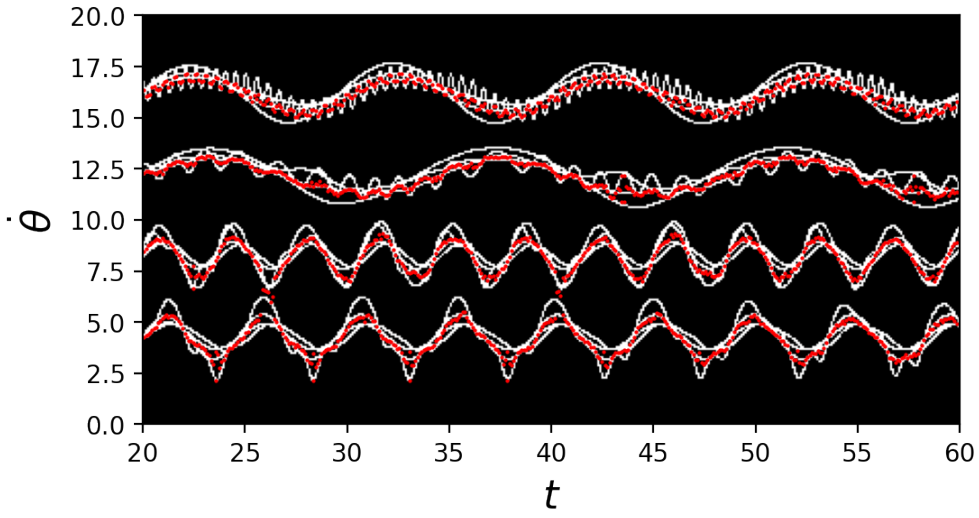

Рис. 5: Центроиды кластеров (выделены красным), распознанных в каждом снимке пространства состояний из серии 400 снимков, представленной в виде столбцов матрицы $\mathfrak{B}$, при значении минимального кластера $M_{\min} = 2$

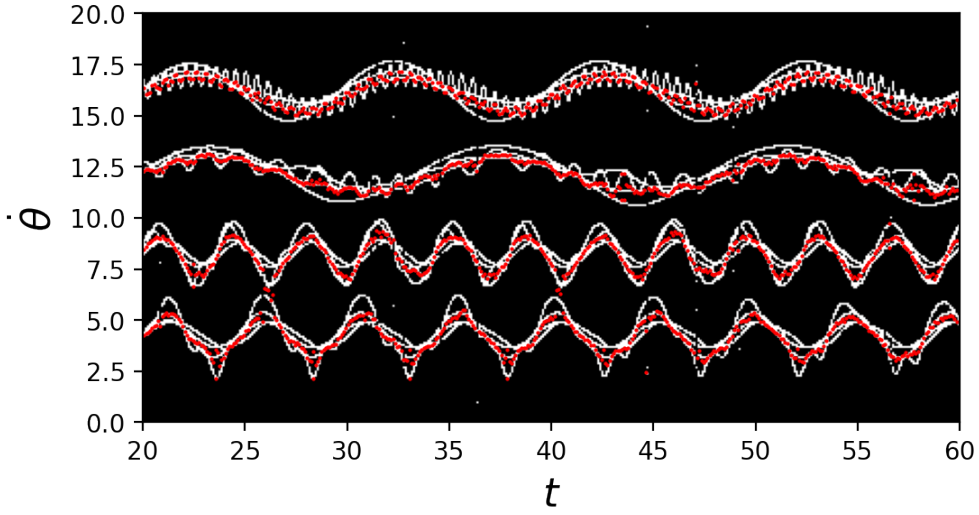

Рис. 6: Центроиды кластеров (выделены красным), распознанных в реконструированных снимках пространства состояний из серии 400 снимков, представленной в виде столбцов матрицы $\widehat{\mathfrak{B}}$, при значении минимального кластера $M_{\min} = 2$

Далее следует определить величину m: пусть она равна целой части от $c \cdot s_0 = 3 * 24.78$, то бишь $m = 74$ (где s_0 — среднее значение разреженности столбцов $\mathfrak{B}$, а c — некоторая константа). Соответствующая

степень сжатия $\gamma_3 = p/m \approx 2.7$. После сжатия и реконструкции по метрике (14) можно сравнить точность нахождения кластеров в оригинальном наборе наблюдений (матрице $\mathfrak{B}$) и в наблюдениях после сжатия-реконструкции (матрице $\widehat{\mathfrak{B}}$). Пусть в качестве выбросов считаются все кластеры, количество элементов в которых меньше 2 ($M_{\min} = 2$). Пусть также всюду далее значение $\delta = 0.8$ — пороговое значение при определении кластеров (если два ненулевых элемента вектора наблюдений располагаются ближе, чем на δ друг от друга, то они относятся к одному кластеру). Результаты сравнения точности представлены в Таблице 3. На Рисунках 5 и 6 обозначены найденные центроиды.

$\epsilon, \%$	
$\mathfrak{B}$	$\widehat{\mathfrak{B}}$
95.74	94.24

Таблица 3: Точность кластеризации при $c = 3$ и размере минимального кластера $M_{\min} = 2$.

6.3. Исследование зависимости точности от степени сжатия

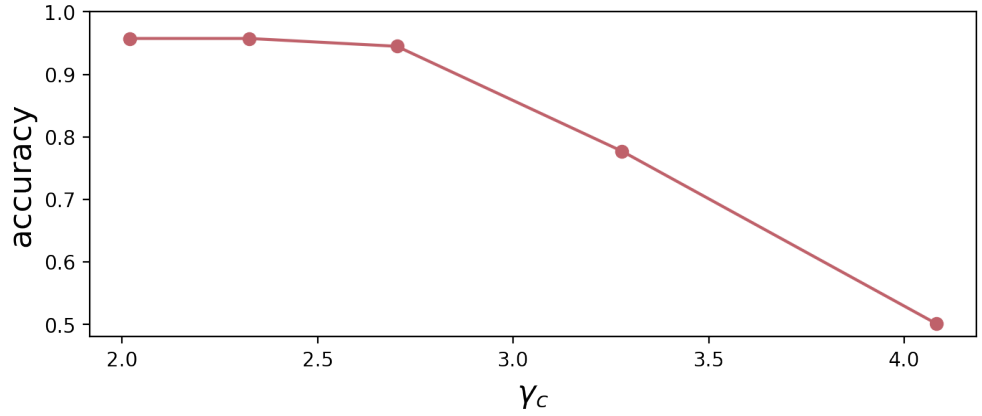

Рис. 7: Точность идентификации кластеров в $\widehat{\mathfrak{B}}$ при $M_{\min} = 2$ и различных значениях степени сжатия γ_c (значения c от 2 до 4 с шагом 0.5)

Точность нахождения кластеров зависит от степени сжатия: чем выше степень сжатия, тем больше потерь, из чего следует меньшая точность распознавания кластеров. На Рисунке 7 представлена зависимость точности определения кластеров в $\widehat{\mathfrak{B}}$ для различных значений γ_c и $M_{\min} = 2$. Видно, что точность монотонно убывает при увеличении степени сжатия.

6.4. Исследование зависимости точности от размера минимального кластера

При размере минимального кластера $M_{\min} = 1$ в качестве отдельного кластера принимаются любые выбросы. В то же время при больших значениях $M_{\min}$ множество центроидов может оказаться пустым. Обе ситуации приводят к низкой точности кластеризации, поэтому важно найти оптимальную величину минимального кластера для максимизации точности. На Рисунке 8 изображена зависимость точности определения кластеров в $\widehat{\mathfrak{B}}$ для различных значений $M_{\min}$ и $\gamma_{2.5} = 3.28$. Из полученного графика следует, что оптимальным значением минимального кластера является 2.

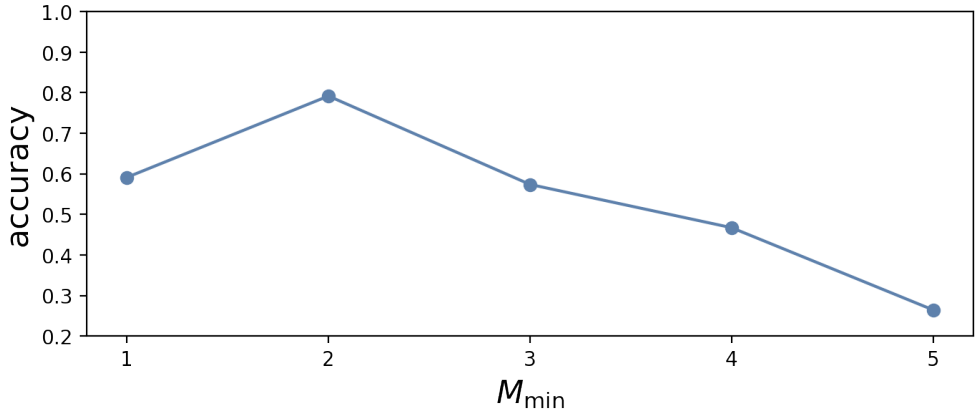

Рис. 8: Точность идентификации кластеров в $\widehat{\mathfrak{B}}$ при $\gamma_{2.5} = 3.28$ и различных значениях минимального кластера $M_{\min}$ (от 1 до 5)

Заключение

В ходе выполнения работы были получены следующие результаты.

1. Выявлены преимущества и недостатки “локального” и “глобального” методов управления, а также двух основных способов моделирования МАС (автоматный и на основе систем дифференциальных уравнений). Проведён анализ метода опознавания по сжатию и модели взаимодействующих осцилляторов (модель Курамото), используемой далее для апробации разработанного алгоритма. Проведено сравнение алгоритмов кластеризации, обоснован выбор иерархического метода.

2. На основе теории динамических систем разработан фреймворк, способный описывать “кластерные потоки” (динамическую кластеризацию, меняющуюся во времени). Продемонстрировано, как в рамках фреймворка можно синтезировать алгоритм кластерного управление агентами.

3. На основе метода Compressive Sensing разработан алгоритм обработки данных о состоянии агентов в киберфизической системе с последующим поиском кластеров. Разработан способ оценки точности алгоритма.

4. Определены требования к программному обеспечению, необходимые для реализации гибко настраиваемого масштабируемого ПО для наблюдения за МАС (в том числе симуляциями) и распознавания в ней кластеров.

5. Выполнена реализация алгоритма кластерного управления с компрессией на языке Python с использованием библиотек NumPy, SciPy, matplotlib. Реализован симулятор МАС для последующей апробации. Ссылка на репозиторий с кодом: [56].

6. На основе модели Курамото разработан пример мультиагентной системы. На данном примере апробирован разработанный и реа-

лизованный алгоритм. При варьировании степени сжатия от 2 до 4 точность падает с 95 до 50 %.

По теоретическим результатам сделан доклад на “Восемнадцатой Национальной конференции по искусственному интеллекту“ (Россия, Москва) и опубликована статья [25]. По результатам разработки алгоритма опознавания по сжатию для MAC опубликована статья [26] в журнале Mathematics с импакт-фактором 1.747 (Q1, Web of Science).

Список литературы

[1] Al-Rifaie Mohammad Majid, Bishop John. Stochastic Diffusion Search Review. — 2013. — 01. — Vol. 4. — P. 155–173.

[2] Application of Improved Multi-Objective Ant Colony Optimization Algorithm in Ship Weather Routing / Guangyu Zhang, Hongbo Wang, Wei Zhao et al. // Journal of Ocean University of China. — 2021. — 02. — Vol. 20. — P. 45–55.

[3] Austin Christopher, Kusumoto Fred. The application of Big Data in medicine: current implications and future directions // Journal of interventional cardiac electrophysiology : an international journal of arrhythmias and pacing. — 2016. — 10. — Vol. 47.

[4] Banait Meenu, Dhok S., Deshmukh R. A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications // IEEE Access. — 2018. — 01. — Vol. PP. — P. 1–1.

[5] Benedetto Dario, Caglioti Emanuele, Montemagno Umberto. On the complete phase synchronization for the Kuramoto model in the mean-field limit // Communications in Mathematical Sciences. — 2014. — 07. — Vol. 13.

[6] Big data analytics in medical imaging using deep learning / Amirhessam Tahmassebi, Anahid Ehtemami, Behshad Mohebali et al. — 2019. — 05. — P. 13.

[7] Candès Emmanuel, Romberg Justin, Tao Terence. Robust Uncertainty Principles : Exact Signal Frequency Information // Information Theory, IEEE Transactions on. — 2006. — 03. — Vol. 52. — P. 489–509.

[8] Chopra Nikhil, Spong M.W. On Synchronization of Kuramoto Oscillators. — Vol. 2005. — 2006. — 01. — P. 3916 – 3922.

[9] A Comprehensive Study on Load Balancing Algorithms in Cloud / Mohona Bandyopadhyay, Manoj Mishra, B. Mishra, Samaresh Mishra. — 2021. — 01. — P. 423–436. — ISBN: 978-981-15-7510-5.

[10] Data Clustering: Algorithms and Its Applications / Jelili Oyelade, Itunuoluwa Isewon, Funke Oladipupo et al. — 2019. — 07. — P. 71–81.

[11] Density-based clustering / Ricardo Campello, Peer Kröger, Joerg Sander, Arthur Zimek // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. — 2019. — 10. — Vol. 10.

[12] Design and Implementation of Multi-Agent Online Auction Systems in Cloud Computing / Hongyan Yu, Srikanta Patnaik, Shenjia Ji et al. — 2021. — 01. — P. 251–269. — ISBN: 9781799853404.

[13] Distribution-Based Cluster Structure Selection / Zhiwen Yu, Xianjun Zhu, Hau-San Wong et al. // IEEE transactions on cybernetics. — 2016. — 06. — Vol. PP.

[14] Donoho David. Compressed Sensing // Information Theory, IEEE Transactions on. — 2006. — 05. — Vol. 52. — P. 1289 – 1306.

[15] Dumaldar Mahesh. A theoretical comparison between the Simplex Method and the Basic Line Search Algorithm // Optimization. — 2014. — 12.

[16] Dynamic Network Bandwidth Resizing for Big Data Applications / Fábio Rossi, Guilherme Rodrigues, Rodrigo Calheiros, Marcelo Conterato. — 2017. — 10.

[17] Falco Mariana, Robiolo Gabriela. A Systematic Literature Review in Multi-Agent Systems: Patterns and Trends. — 2019. — 09. — P. 1–10.

[18] Fradkov Alexander. Cybernetical Physics. — Springer, 2007. — ISBN: 978-3-540-46277-4.

[19] Gazi Veysel, Fidan Baris. Coordination and Control of Multi-agent Dynamic Systems: Models and Approaches. — Vol. 4433. — 2007. — 07. — P. 71–102.

[20] Gholizadeh Nahid, Saadatfar Hamid, Hanafi Nooshin. K-DBSCAN: An improved DBSCAN algorithm for big data // The Journal of Supercomputing. — 2020. — 11. — P. 1–22.

[21] Giordani Paolo, Ferraro Maria, Martella Francesca. Hierarchical Clustering. — 2020. — 08. — P. 9–73. — ISBN: 978-981-13-0552-8.

[22] Goldstein Jeffrey. Emergence as a Construct: History and Issues // Emergence. — 1999. — 03. — Vol. 1. — P. 49–72.

[23] Grandmaster level in StarCraft II using multi-agent reinforcement learning / Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki et al. // Nature. — 2019. — 11. — Vol. 575.

[24] Granichin Oleg, Pavlenko Dmitry. Randomization of data acquisition and ℓ_1-optimization // Autom Remote Control. — 2010. — Vol. 71. — P. 2259–2282.

[25] Granichin Oleg, Uzhva Denis. Invariance Preserving Control of Clusters Recognized in Networks of Kuramoto Oscillators. — 2020. — 11.

[26] Granichin Oleg, Uzhva Denis, Volkovich Zeev. Cluster Flows and Multiagent Technology // Mathematics. — 2020. — 12. — Vol. 9. — P. 22.

[27] Graphical View of Quick Simplex Method to Solve Linear Programming Problem / Nalini Vaidya, Smita Pidurkar, Manoj Shanti, Satyajit Uparkar. — 2020. — 06. — P. 5694–5710.

[28] Gravitational search algorithm for determining the optimal kinetic parameters of propane pre-reforming reaction / Leniza Enikeeva, Dmitriy Potemkin, Sergey Uskov et al. // Reaction Kinetics, Mechanisms and Catalysis. — 2021. — 01. — Vol. 132.

[29] Investigation on performance of particle swarm optimization (PSO) algorithm based fuzzy inference system (PSOFIS) in a combination of CFD modeling for prediction of fluid flow / Meisam Babanezhad, Iman Behroyan, Ali Taghvaie Nakhjiri et al. // Scientific Reports. — 2021. — 01. — Vol. 11.

[30] Jadbabaie Ali, Motee Nader, Barahona Mauricio. On the stability of the Kuramoto model of coupled nonlinear oscillators. — 2005. — 05. — P. 4296 – 4301 vol.5.

[31] Karna Ashutosh, Gibert Karina. Automatic identification of the number of clusters in hierarchical clustering // Neural Computing and Applications. — 2021. — 03.

[32] Kotwal Tejas, Jiang Xin, Abrams Daniel. Connecting the Kuramoto Model and the Chimera State // Physical Review Letters. — 2017. — 12. — Vol. 119.

[33] The Kuramoto model: A simple paradigm for synchronization phenomena / Juan Acebron, Luis Bonilla, Conrado Pérez-Vicente et al. // Reviews of Modern Physics. — 2005. — 04. — Vol. 77.

[34] Learning a Compressed Sensing Measurement Matrix via Gradient Unrolling / S. Wu, A. Dimakis, S. Sanghavi et al. // ICML. — 2019.

[35] Leonard Naomi Ehrich. Multi-Agent System Dynamics: Bifurcation and Behavior of Animal Groups // IFAC Proceedings Volumes. — 2013. — Vol. 46, no. 23. — P. 307–317. — 9th IFAC Symposium on Nonlinear Control Systems. Access mode: https://www.sciencedirect.com/science/article/pii/S1474667016316779.

[36] Li Hao, Li Yuejiang, Zhao H. Vicky. Modeling Decision Process in Multi-Agent Systems: A Graphical Markov Game based Approach // 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). — 2020. — P. 197–204.

[37] Lodge Paul. Leibniz's Mill Argument Against Mechanical Materialism Revisited // Ergo, an Open Access Journal of Philosophy. — 2014. — Vol. 1, no. 3.

[38] Lu Wenlian, Atay Fatihcan. Stability of Phase Difference Trajectories of Networks of Kuramoto Oscillators with Time-Varying Couplings and Intrinsic Frequencies // SIAM Journal on Applied Dynamical Systems. — 2018. — 01. — Vol. 17. — P. 457–483.

[39] Luo Shengping, Ye Dan. Cluster consensus control of linear multi-agents systems under directed topology with general partition // IEEE Transactions on Automatic Control. — 2021. — 03. — Vol. PP. — P. 1–1.

[40] Measurements of Ξ^- and $\overline{\Xi}^+$ production in proton–proton interactions at $\sqrt{s_{NN}} = 17.3$ GeV in the NA61/SHINE experiment / Antoni Aduszkiewicz, Evgeny Andronov, Tome Anticic et al. // The European Physical Journal C. — 2020. — 09. — Vol. 80. — P. 833.

[41] Measurements of $\pi^{\pm}$, $K^{\pm}$, p and $\bar{p}$ spectra in ^{7}Be+^{9}Be collisions at beam momenta from 19A to 150A GeV/c with the NA61/SHINE spectrometer at the CERN SPS / Abhipsa Acharya, Haradhan Adhikary, A. Aduszkiewicz et al. // The European Physical Journal C. — 2021. — 01. — Vol. 81.

[42] Montbrió Ernest, Pazó Diego, Schmidt Jürgen. Time delay in the Kuramoto model with bimodal frequency distribution // Physical review. E, Statistical, nonlinear, and soft matter physics. — 2006. — 12. — Vol. 74. — P. 056201.

[43] Mughnyanti M, Efendi Sahrul, Zarlis M. Analysis of determining centroid clustering x-means algorithm with davies-bouldin index evaluation // IOP Conference Series: Materials Science and Engineering. — 2020. — 01. — Vol. 725. — P. 012128.

[44] Nemirovski Arkadi, Todd Michael. Interior-point methods for optimization // Acta Numerica. — 2008. — 05. — Vol. 17. — P. 191 – 234.

[45] Nesterov Yu, Todd Michael. Primal-Dual Interior-Point Methods for Self-Scaled Cones // SIAM Journal on Optimization. — 1998. — 02. — Vol. 8. — P. 324–364.

[46] Proskurnikov Anton, Granichin Oleg. Evolution of clusters in large-scale dynamical networks // Cybernetics and Physics. — 2018. — 11. — Vol. 7, no. 3. — P. 102–129.

[47] Rashid Salman, Razak Shukor. Big Data Challenges in 5G Networks. — 2019. — 07. — P. 152–157.

[48] Requicha Aristides. Nanorobots, NEMS, and nanoassembly // Proceedings of the IEEE. — 2003. — 12. — Vol. 91. — P. 1922 – 1933.

[49] Ribarics Pal. Big Data and its impact on agriculture // Ecocycles. — 2016. — 07. — Vol. 2. — P. 33–34.

[50] Road traffic management based on self-load-balancing approach / Ahmed Adnane, Mohamed Sadik, Saida Talal et al. // International Journal for Simulation and Multidisciplinary Design Optimization. — 2016. — 01. — Vol. 7. — P. A9.

[51] Rosales Rómer, Schmidt Mark, Fung Glenn. Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches. — 2007. — 09.

[52] Sadilek Maximilian, Thurner Stefan. Physiologically motivated multiplex Kuramoto model describes phase diagram of cortical activity // Scientific Reports. — 2014. — 09. — Vol. 5.

[53] Shehadeh Hisham. A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization // Neural Computing and Applications. — 2021. — 03.

[54] Silva Elisabete. DG-ABC: An Integrated Multi-Agent and Cellular Automata Urban Growth Model. — 2014. — 01. — P. 57–92. — ISBN: 9781466643505.

[55] Singh Satya. Study of K-Means and Enhanced K-Means Clustering Algorithm. // International Journal of Research in Computer Science. — 2021. — 03. — Vol. Vol. 4 Issue 10. — P. p103–107. 5p.

[56] Source Code of Simulation. — https://github.com/denisuzhva/clusterFlow.

[57] Stability Conditions for Cluster Synchronization in Networks of Heterogeneous Kuramoto Oscillators / Tommaso Menara, Giacomo Baggio, Danielle Bassett, Fabio Pasqualetti // IEEE Transactions on Control of Network Systems. — 2019. — 03. — Vol. 7. — P. 302–314.

[58] Synchronization in a semiclassical Kuramoto model / Ignacio Hermoso de Mendoza Naval, Leonardo Pachón, Jesus Gómez-Gardeñes, David Zueco // Physical review. E, Statistical, nonlinear, and soft matter physics. — 2014. — 11. — Vol. 90. — P. 052904.

[59] Szabo Zsuzsanna, Kovacs Marta. On Interior-Point Methods and Simplex Method in Linear Programming // Analele Stiintifice ale Universitatii Ovidius Constanta, Seria Matematica. — 2003. — 01. — Vol. 11.

[60] Tharwat Alaa, Schenck Wolfram. A conceptual and practical comparison of PSO-style optimization algorithms // Expert Systems with Applications. — 2021. — Vol. 167. — P. 114430. — Access mode: https://www.sciencedirect.com/science/article/pii/S0957417420310939.

[61] Vamvoudakis Kyriakos G., Jagannathan S. Control of Complex Systems: Theory and Applications. — 2016. — 07.

[62] Wooldridge Michael. An Introduction to Multiagent Systems. — 2 edition. — Chichester, UK : Wiley, 2009. — ISBN: 978-0-470-51946-2.

[63] Zhang Xiangfeng, Wang Yanmei. Research on intelligent medical big data system based on Hadoop and blockchain // EURASIP Journal on Wireless Communications and Networking. — 2021. — 01. — Vol. 2021.

Приложение

Перед доказательством Теоремы 1 докажем следующую лемму.

Лемма 1. *Рассмотрим мультиагентную сеть, соответствующую модели* (8) *с нулевым мезо-управлением* ($U_i(t) = \mu_i \mathcal{F}_\alpha(t, \overline{x}_\alpha(t)) = 0$) *и графу* $\mathcal{G}$ *с матрицей связности* Υ. *Пусть* $t \in T$, *вывод* $z_i(t) = \dot{\theta}_i(t)$ *и* $\Delta_{ij}(t) = |z_i(t) - z_j(t)|$. *Следующие условия достаточны для достижения* $(0, 0)$*-синхронизации в этой сети:*

1. *Для* $i, j \in \mathcal{M}_\alpha$ *таких что* $w_j - w_i \geq 0$

$$w_j - w_i \leq \rho \sin\left(\frac{\Delta\theta_{ji}}{2}\right) \sum_{l=1}^{N} [\Upsilon_{il} + \Upsilon_{jl}], \tag{17}$$

где $\sin\left(\frac{\Delta\theta_{ji}}{2}\right) = 1$ *в случае* $\Upsilon_{ij} = \Upsilon_{ji} = 0$; *иначе*

$$\sin\left(\frac{\Delta\theta_{ji}}{2}\right) = \max\left\{\sqrt{1 - (\Gamma_i(j))^2}, \sqrt{1 - (\Gamma_j(i))^2}, \frac{\sqrt{2}}{2}\right\}, \tag{18}$$

где

$$\Gamma_i(j) = \frac{-d_i(j) + \sqrt{(d_i(j))^2 + 8(\Upsilon_{ij} + \Upsilon_{ji})^2}}{4(\Upsilon_{ij} + \Upsilon_{ji})}. \tag{19}$$

2. *Для* $i \in \mathcal{M}_\alpha$, $j \in \mathcal{M}_\beta$, $\alpha \neq \beta$

$$|w_i - w_j| > 0. \tag{20}$$

3. *Граф* $\mathcal{G}$ *является сильно-связным.*

Доказательство 1. *С целью упрощения нотации всюду далее в доказательстве мы пишем* $\dot{\theta}_i(t)$ *как* $\dot{\theta}_i$ *(то же самое относится к* θ_i *и* Δ_{ij}*). Следуя Определению 5,* $\Delta_{ij} = \left|\dot{\theta}_i - \dot{\theta}_j\right| = 0$ $\forall i, j \in \mathcal{M}_\alpha$ *и* $\Delta_{ij} > 0$ $\forall i \in \mathcal{M}_\alpha, j \in \mathcal{M}_\beta$ $\alpha \neq \beta$. *Поскольку ситуация, когда* i *и* j *– из разных кластеров, доказывается проще, то мы рассматриваем ее в первую очередь. Подставив вместо* $\dot{\theta}_i$ *правую часть модели* (8), *получаем сле-*

дующее выражение Δ_{ij}*:*

$$\Delta_{ij} = \left| w_i - w_j + \rho \left(\sum_{l=1}^{N} \Upsilon_{il} \sin(\theta_l - \theta_i) - \sum_{l=1}^{N} \Upsilon_{jl} \sin(\theta_l - \theta_j) \right) \right| > 0. \quad (21)$$

Полагая аргументы синусов в (21) *равными 0, получаем достаточное условие на собственные частоты. Далее мы рассматриваем ситуацию* $i, j \in \mathcal{M}_\alpha$*. Пусть левая часть* (21) *строго равна нулю; предположим, что* $w_j - w_i \geq 0$ *без потери общности. Пусть* E *– функционал*

$$\begin{aligned} E = \sum_{l=1}^{N} \Upsilon_{il} \sin(\theta_l - \theta_i) - \sum_{l=1}^{N} \Upsilon_{jl} \sin(\theta_l - \theta_j) = \\ = (\Upsilon_{ij} + \Upsilon_{ji}) \sin(\theta_j - \theta_i) + \sum_{l=1, l\neq j}^{N} \Upsilon_{il} \sin(\theta_l - \theta_i) - \\ - \sum_{l=1, l\neq i}^{N} \Upsilon_{jl} \sin(\theta_l - \theta_j), \end{aligned} \quad (22)$$

который мы желаем максимизировать. Необходимые условия (первого порядка) для максимизации E

$$\frac{\partial E}{\partial \theta_i} = -(\Upsilon_{ij} + \Upsilon_{ji}) \cos(\theta_j - \theta_i) - \sum_{l=1, l\neq j}^{N} \Upsilon_{il} \cos(\theta_l - \theta_i) = 0, \quad (23)$$

$$\frac{\partial E}{\partial \theta_j} = (\Upsilon_{ij} + \Upsilon_{ji}) \cos(\theta_j - \theta_i) + \sum_{l=1, l\neq i}^{N} \Upsilon_{jl} \cos(\theta_l - \theta_j) = 0, \quad (24)$$

$$\frac{\partial E}{\partial \theta_l} = \Upsilon_{il} \cos(\theta_l - \theta_i) - \Upsilon_{jl} \cos(\theta_l - \theta_j) = 0. \quad (25)$$

Для того чтобы нижеследующие рассуждения были истинными, допустим $\exists l_1 : \Upsilon_{il_1} = 1$ *и* $\exists l_2 : \Upsilon_{jl_2} = 1$*. Поскольку* i *и* j *произвольны, граф* $\mathcal{G}$ *оказывается таким образом сильно-связным. Рассмотрим случай* $\Upsilon_{ij} = \Upsilon_{ji} = 0$*. Тогда косинусы во вторых слагаемых уравнений* (23)

and (24) *также равны 0, из чего следует*

$$E_{\max} = \sum_{l=1}^{N} [\Upsilon_{il} + \Upsilon_{jl}]. \tag{26}$$

Теперь пусть $\Upsilon_{ij} + \Upsilon_{ji} \geq 1$. *Рассмотрим уравнение* (25). *В случае* $\Upsilon_{il} = \Upsilon_{jl} = 0$ *получается, что* $E_{\max}$ *имеет вид как в* (26). *Если же* $\Upsilon_{il} = 1$ *и* $\Upsilon_{jl} = 0$, *тогда, следуя* (23), $\cos(\theta_j - \theta_i) = 0$. *То же самое истинно если* $\Upsilon_{il} = 0$ *и* $\Upsilon_{jl} = 1$, *следуя* (24). *Самая нетривиальная ситуация – когда* $\Upsilon_{il} = \Upsilon_{jl} = 1$. *Используя* (25), *получаем, что либо* $\theta_l = \frac{\theta_i + \theta_j}{2}$, *либо* $\theta_i = \theta_j = 0$. *Однако, из второго следует, что* $E \equiv 0$, *что не соответствует цели максимизации* E. *Таким образом, подставляем* $\theta_l = \frac{\theta_i + \theta_j}{2}$ *в* (23) *и используем выражение* $\cos(2x) = 2\cos^2(x) - 1$:

$$2(\Upsilon_{ij} + \Upsilon_{ji}) \cos^2\left(\frac{\theta_j - \theta_i}{2}\right) - (\Upsilon_{ij} + \Upsilon_{ji}) + \cos\left(\frac{\theta_j - \theta_i}{2}\right) \sum_{l=1, l\neq j}^{N} \Upsilon_{il} = 0. \tag{27}$$

После решения квадратного уравнения (27) *получаем два решения, единственное из которых (со знаком "плюс") удовлетворяет следующему условию:* $\cos\left(\frac{\theta_j - \theta_i}{2}\right) \in [-1, 1]$. *Выбранное решение мы обозначаем* $\Gamma_j(i)$ *– оно соответствует* (19). *Аналогичным образом получаем решения* (24). *И снова подходит лишь решение со знаком "плюс'', обозначаемое как* $\Gamma_i(j)$. *Пусть оптимальное значение* $(\theta_j - \theta_i)_{opt} = \Delta\theta_{ji}$ *как в* (18), *где первые два варианта получены из выражения* $\sin(\arccos(x)) = \sqrt{1 - x^2}$, *в то время как последний выведен из* (25) *в случае если один из коэффициентов равен 1, а другой равен 0. Наконец, подставим* $\Delta\theta_{ji}$ *и* $\theta_l = \frac{\theta_i + \theta_j}{2}$ *в* (22):

$$E_{\max} = \sin\left(\frac{\Delta\theta_{ji}}{2}\right) \sum_{l=1}^{N} [\Upsilon_{il} + \Upsilon_{jl}], \tag{28}$$

что завершает доказательство.

С целью упрощения нотации $\mathcal{F}_\alpha = \mathcal{F}_\alpha(t, \overline{x}_\alpha)$. Далее следует доказательство Теоремы 1.

Доказательство 2. *Достаточные условия можно получить из Леммы 1 с помощью подстановки мезоскопического управления U_i в Δ_{ij}:*

$$\Delta_{ij} = \Big| w_i - w_j + \mu_i \mathcal{F}_\alpha - \mu_j \mathcal{F}_\beta + \\ + \rho \cdot \left(\sum_{l=1}^{N} \Upsilon_{il} \sin(\theta_l - \theta_i) - \sum_{l=1}^{N} \Upsilon_{jl} \sin(\theta_l - \theta_j) \right) \Big|. \tag{29}$$

Для начала, предположим, что $i, j \in \mathcal{M}_\alpha$. Следуя тем же рассуждениям, что в Доказательстве Леммы 1, уравнение (29) *равно 0, из чего получается* (9)*. Теперь пусть $i \in \mathcal{M}_\alpha, j \in \mathcal{M}_\beta$ $\alpha \neq \beta$, таким образом $\Delta_{ij} > 0$ в уравнении* (29)*. Полагая синусы равными нулю, как в Лемме 1, получается желаемое условие на мезоскопическое управление, что завершает доказательство.*