МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

На правах рукописи

МАХРОВ СТАНИСЛАВ СТАНИСЛАВОВИЧ

ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ МЕХАНИЗМОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ КЛАСТЕРИЗАЦИИ УЗЛОВ И МАРШРУТИЗАЦИИ ДАННЫХ В БЕСПРОВОДНЫХСЕНСОРНЫХ СЕТЯХ

Специальность 05.12.13 – «Системы, сети и устройства телекоммуникаций»

ДИССЕРТАЦИЯ

на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, доцент
Ерохин Сергей Дмитриевич

МОСКВА– 2015

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы.Беспроводные сенсорные сети (БСС) – это самоорганизующиеся, распределенные, масштабируемые сети, состоящие из множества автономных сенсоров (сенсорных узлов), объединенных посредством радиоканала. Узлы являются автономными в отношении электропитания, для поддержки работоспособности сети не требуется обслуживающий персонал, а сеть может перестраиваться с течением времени.

В настоящее время как этап глобальной информатизации и становления информационного общества, происходит активное развитие единой информационной среды[83]. Интернет вещей (InternetOfThings) является основной концепций данного развития, согласно которой планируется практически каждое бытовое устройство оснастить подключением к сети Интернет [81]. При этом устройства будут называться вещами - «предметами физического или информационного мира, которые могут быть идентифицированы и интегрированы в сети связи», согласно с определением Международного союза электросвязи (МСЭ) в рекомендации Y.2069 [55].

К настоящему моменту часть узлов из общего числа подключенных к Интернету вещей являются узлами БСС. Это обусловлено тем, что последние являются одним из направлений развития Интернета вещей, предоставляя широкие возможности для интеграции в различные процессы.

Сферы применения БСС различны: мониторинг промышленности и производства, сети технологического контроля, здравоохранение, военные технологии [90], мониторинг окружающей среды, системы «интеллектуальный дом», логистика и навигация и многие другие [96]. БСС позволяют производить мониторинг и контроль физических параметров или объектов на разных уровнях для решения различных задач [36].

Степень разработанности темы.

Среди российских исследователей БСС наиболее известны работы следующих ученых: А.Е. Кучерявого, Е.А. Кучерявого, А.С. Лебедева, В.М.

Вишневского, Г.Ф. Гайкович, С.С. Баскакова, В.И. Оганова, А.С.Дмитриева, Л.В.Кузьмина, В.Ю. Юркина, Т.И. Мохсени, С.В. Трифоновой, Я.А. Холодова, Л.С. Воскова.

Вотношении зарубежных исследователейБСС, можно выделить труды W. Dargie, ,K. Sohraby, D. Minoli, T. Znati, V. Peiris, W.R.Heinzelman, M. Magno, D. Boyle, D.Brunelli, B. O'Flynn, C. Poellabauer,E. Popovici, L.Benini, D.Silva, M.Ghanem, Y.Guo.

Из исследований протоколов маршрутизации наиболее известны работы M.J. Handy, M. Haase, D. Timmermann, Y. Yu, R. Govindan, D. Estrin, B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, M. Zorzi, R. R. Rao, V. Rodoplu, T. H. Meng, L. Li, J. Y. Halpern, W. R. Heinzelman, J. Kulik, C. Intanagonwiwat, D. Braginsky, Y. Yao, J. Gehrke, N. Sadagopan, B. Krishnamachari, A. Helmy, A. Boukerche, X. Cheng, J. Linus, S. Lindsey, C.S. Raghavendra, O. Younis, S. Fahmy.

В последнее время проводится множество исследований области иерархических протоколов маршрутизации, так как данный класс протоколов является энергоэффективным на основании последних исследований в этой области.

Вместе с тем, существует несколько ключевых проблем при самоорганизации и маршрутизации данных в БСС:

1) При самоорганизации часть узлов может быть не задействована и оказаться вне состава сети.
2) Многие протоколы маршрутизации не способны адаптироваться к изменению местоположения базовой станции (БС).
3) В«плоских» протоколах при передаче данных от узла А до узла В расходуется энергия всех промежуточных узлов, что негативно сказывается на времени жизни всей сети.
4) Во многих иерархических протоколах часть узлов, либо все узлы, включенные в состав кластера, часто не могут передавать данные между собой, поскольку при кластеризации используется не радиовидимость

узлов, а их координаты, полученные посредством GPS/ГЛОНАСС. Кроме того, в случае, если для определения соседних узлов используется наличие радиосигнала, то в таких протоколах, как правило, при формировании кластеров не принимается во внимание наличие/отсутствие и уровень мощности радиосигнала всех остальных узлов, что ведет к неправильному формированию кластеров.

5) Проблема масштабируемости сети.

Для иерархических протоколов с целью построения иерархии, одной из основных задач является кластеризация узлов, поскольку от нее зависит масштабируемость и эффективность работы сети. Кластеризацией в БСС называется разделение узлов сети на отдельные группы (кластеры), во главе каждой из которых назначается главный кластерный узел (ГКУ), осуществляющий маршрутизацию данных между узлами кластера и передающий агрегированные данные на БС[80].

В качестве инструментов кластеризации в иерархических протоколах маршрутизации используются различные способы. В данной диссертационной работе в качестве способа кластеризации предлагается использовать механизмы искусственного интеллекта, выраженные в виде математической модели – искусственной нейронной сети (ИНС) и предлагается протокол маршрутизации, реализующий возможности такого подхода.

Цельюдиссертационной работы является повышение эффективности самоорганизации БСС и маршрутизации данных в ней посредством использования механизмов искусственного интеллекта нейронной сети.

Поставленная цель определила необходимость решения следующих **задач**:

- анализ существующих протоколов маршрутизации для БСС;
- исследование топологий, моделей связности узлов;
- исследование в области определения эффективной модели связности узлов;

- исследование существующих архитектур нейронных сетей на предмет эффективности их применения в области БСС;
- разработка способа кластеризации БСС с помощью нейронной сети;
- моделирование разработанного способа кластеризации БСС с помощью нейронной сети;
- разработка протоколамаршрутизации данных, использующего нейронную сеть для самоорганизации;
- моделирование и сравнение разработанного протокола с другими протоколами, используемыми в БСС.

Объектом исследования является беспроводная сенсорная сеть.

Предметом исследования являются алгоритмы и методы кластеризации и маршрутизации данных в беспроводной сенсорной сети.

Научная новизна диссертационной работы.

В результате исследований получены следующие новые научные результаты:

1. Предложено использовать искусственные нейронные сети для кластеризации беспроводных сенсорных сетей, что позволяет использовать различные параметры (уровень радиовидимости, уровень остаточной энергии, приоритет узлов и т.д.) для кластеризации узлов и повысить время работы узлов сети.
2. Предложена матрица радиовидимости, являющаяся математическим описанием связности узлов сети и радиовидимости каждого узла по отношению ко всем остальным узлам сети.
3. Исследована эффективность кластеризации с помощью нейронной сети – Самоорганизующейся карты Кохонена, обучаемой по Конструктивному методу.
4. Разработан способ нейросетевой кластеризации беспроводной сенсорной сети, основанный на архитектуре сети Кохонена, обучаемой по Конструктивному методу.

5. Разработан матричный способ кластеризации беспроводной сенсорной сети.
6. Разработан протокол маршрутизации данных для беспроводных сенсорных сетей, кластеризованных с использованием нейронных сетей, что позволяет повысить жизненный цикл сети на 27% по сравнению с существующими протоколами маршрутизации данных БСС.

Методы исследования. Для решения поставленных задач в работе используются методы искусственных нейронных сетей, теории графов, теории сетей связи, искусственного интеллекта, математического и компьютерного моделирования.

Теоретическую основу исследования составили работы по развитию информационного общества А.С. Аджемова, А.Е. Кучерявого, Е.А. Кучерявого, моделированию О.И. Шелухина, C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, A. Durresi, анализу данных посредством искусственных нейронных сетей Т. Кохонена, Ф. Розенблата, Д.В. Постарнака, А.С. Баталова, К.В. Воронцова, Ф. Уоссермена, протоколам маршрутизации БСС А.Е. Кучерявого, Е.А. Кучерявого, Л.С. Воскова,M.J. Handy, M. Haase, D. Timmermann, Y. Yu, R. Govindan, D. Estrin, B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, M. Zorzi, R. R. Rao.

Практическая значимость и реализация результатов работы. Выполненные в диссертационной работе исследования, а также предложенные способы, могут быть использованы для самоорганизации БСС и маршрутизации данных, позволяя включить в сеть максимальное количество узлов и увеличить время жизни всей сети. Нейросетевой и матричный способы кластеризации позволяют формировать кластеры из узлов БСС на основании матрицы радиовидимости, которая является аналитическим представлением графа, описывающего связи между всеми узлами сети. Наличие матрицы радиовидимости в качестве входных данных, позволяет при формировании кластеров учитывать знания обо всех узлах сеть для того, чтобы корректно

выделить кластеры. Ориентированность способов кластеризации на использование в иерархических протоколах позволит оптимизировать использование энергии, поскольку в результате проведенных исследования, выявлено, что иерархические протоколы являются наиболее эффективными за счет агрегации и сжатия данных только на определенных узлах – главных кластерных узлах (ГКУ).

Разработанный протокол маршрутизации может использовать нейросетевой или матричный способы кластеризации. Благодаря этому, протокол позволяет оптимизировать передачу данных в сети, повысить ее время жизни и живучесть. Иерархическая направленность протокола обеспечивает высокую масштабируемость сети (до 10 000 узлов и более) и позволяет использовать мобильную базовую станцию.

Протокол, разработанный в рамках данной диссертационной работы, может быть использован в новом классе сенсорных управленческих сетей (СУС), описанных в рекомендации МСЭ-Т Y.2222 [56], а также благодаря его ориентированности на БСС, может быть использован в устройствах, разрабатываемых согласно концепции Интернета вещей в соответствии с рекомендацией МСЭ-Т Y.2069 [55].

Основные результаты диссертационной работы использованы в практической деятельности органа исполнительной власти Правительства Москвы – Департамента городского имущества г.Москвы и в учебном процессе ФГОБУ ВПО МТУСИ.

Соответствие диссертационной работы паспорту научной специальности.

Диссертационная работа содержит исследование вопросов создания новых методов обеспечения эффективности работы беспроводных сенсорных сетей и соответствует следующим пунктам паспорта научной специальности 05.12.13: «3. Разработка эффективных путей развития и совершенствования архитектуры сетей и систем телекоммуникаций и входящих в них устройств»,

«6. Развитие операционной среды, формирующей единство, синергетичность и адаптивность телекоммуникаций», «11. Разработка научно-технических основ технологии создания сетей, систем и устройств телекоммуникаций и обеспечения их эффективного функционирования», «12. Разработка методов эффективного использования сетей, систем и устройств телекоммуникаций в различных отраслях народного хозяйства».

Основные положения, выносимые на защиту:

1. Использование механизма искусственных нейронных сетей позволяет эффективно кластеризовать узлы беспроводных сенсорных сетей с учетом множества различных параметров - уровень радиовидимости, уровень остаточной энергии, приоритет узлов и т.д.
2. Использование матрицы радиовидимости позволяет проводить кластеризацию с учетом уровня радиосигналов соседних устройств.
3. Предложенный способ кластеризации узлов беспроводных сенсорных сетей с использование искусственной нейронной сети, обучаемой по Конструктивному методу, позволяет определить принадлежность узла к кластеру точнее, чем способы, используемые в традиционных протоколах маршрутизации, что позволяет увеличить жизненный цикл сети.
4. Разработанный протокол энергетических расстояний нейросетевой кластеризации позволяет организовать передачу данных в беспроводных сенсорных сетях, кластеризованных с использованием ИНС, позволяет повысить жизненный цикл сети на 27% по сравнению с одним из самых эффективных протоколов TEEN.

Степень достоверности и апробация результатов работы. Достоверность результатов обеспечивается адекватностью используемых математических методов, верификацией математической модели нейронной сети путем компьютерного моделирования.

Основные результаты диссертационной работы докладывались и обсуждались на Международной научно-технической конференции

«INTERMATIC-2013» МИРЭА (г. Москва, 2013), на 7-ой и 8-ой Международной отраслевой научно-технической конференции «Технологии информационного общества» МТУСИ (г. Москва, 2013-2014), на XIV Всероссийской выставке научно-технического творчества молодежи (НТТМ), ВДНХ (г. Москва, 2014), на Всероссийской научно-технической конференции, посвященной теоретическим и прикладным проблемам развития и совершенствования автоматизированных систем управления специального назначения «НАУКА И АСУ - 2014» МТУСИ (г. Москва, 2014).

Личный вклад. Все основные научные положения и выводы, составляющие содержание диссертации, полученысоискателем самостоятельно. Теоретические и практические исследования, а также вытекающие из них выводы и рекомендации, получены автором лично.

Публикации. По теме диссертации опубликовано 15 печатных работ, в том числе 4 работы в ведущих рецензируемых научных журналах и изданиях, внесенных в перечень журналов и изданий, утвержденных ВАК. Имеется 2 свидетельства о регистрации программного обеспечения, а также в настоящее время проходит регистрацию патент на изобретение – «Нейросетевой способ кластеризации беспроводной сенсорной сети».

ГЛАВА 1. Обзор протоколов маршрутизации и моделей связности в беспроводных сенсорных сетях

1.1. Маршрутизация данных в БСС

Сенсорный узел - минимальная единица БСС, с помощью которой образуется сеть, при развертывании таких узлов в некоторой целевой области и установлении между ними радиоканала. Сенсорные узлы с помощью датчиков (сенсоров) производят сбор информации из внешней среды и передают её, как правило, на базовую станцию (БС), в качестве которой может служить персональный компьютер, ноутбук, планшет или любая другая рабочая станция [13].

Аппаратная архитектура сенсорного узла изображена на рисунке 1.1. Сенсорный узел (узел сети,сенсор), содержит датчик, воспринимающий данные от внешней среды (собственно сенсор), микроконтроллер, память, радиопередатчик, автономный источник питания и иногда исполнительные механизмы. Возможна также передача управляющих воздействий от узлов сети к внешней среде [84].

Схематичносенсорный узел можно разделить на 4 блока: сенсорный блок; вычислительный блок; коммуникационный блок (приемопередатчик); блок питания [52, 92].

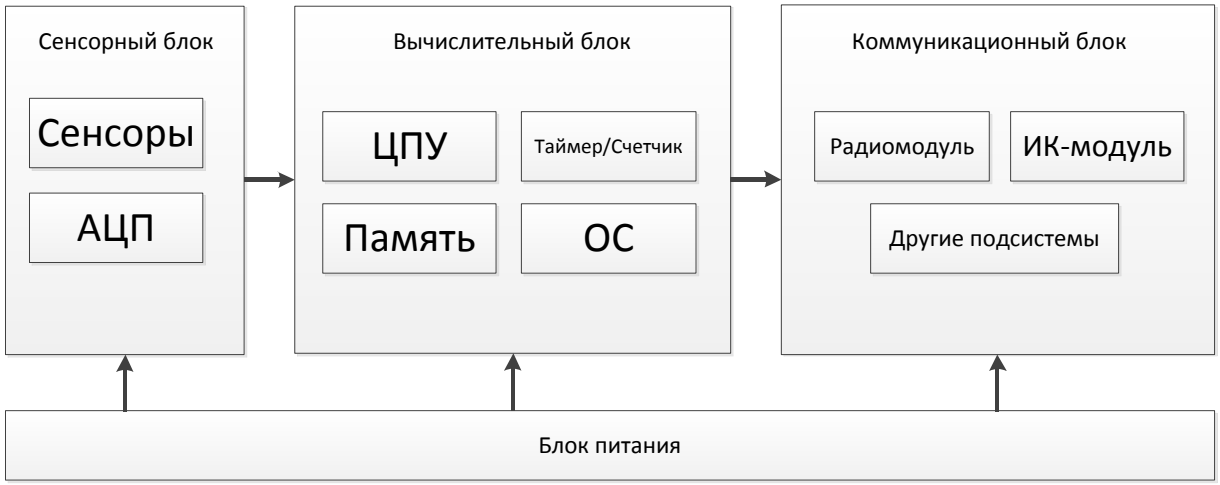


Рисунок 1.1 Аппаратная архитектура сенсорного узла БСС

Главными отличиями БСС от типичных беспроводных сетей, являются следующие свойства:

- Самоорганизация – возможность автоматического построения сети с именованием узлов определенными идентификаторами, например IP-адрес, символьное имя и др. достигается за счет совокупной работы протоколов маршрутизации, пригодных для работы в БСС и встроенного программного обеспечения (ПО) в узлах сети. На протоколы маршрутизации возлагается не только задача составления таблицы маршрутизации на основании заданных критериев определения оптимального маршрута, но и обнаружение соседних беспроводных узлов для осуществления построения сети [67].
- Надежность и отказоустойчивость, основанная на самоорганизации – данное свойство вытекает из первого, поскольку, благодаря самоорганизации, сеть может переконфигурироваться в случае выхода из строя одного или нескольких сенсоров.
- Автономность – наличие собственных автономных источников питания.
- Масштабируемость – возможность увеличения количества узлов сети.
- Связность – возможность перемещения узлов сети в пространстве, не нарушая при этом логической связанности сети.

1.2. Анализ существующих протоколов маршрутизации

Использование протокола, который бы эффективно обеспечивал работу БСС в заданных условиях, при определенных параметрах и в зависимости от типа данных, сбор которых осуществляется, является важной проблемой на современном этапе развития информационного общества [28, 47, 82]. Кроме того, как было отмечено ранее, одной из важных задач в этой связи является обеспечение максимально долгого времени жизни и безотказной работы БСС, решение которой возлагается также и на протокол маршрутизации, под

управлением которого функционирует БСС [4]. Поэтому эффективность, адекватность получаемых данных жизненный цикл БСС напрямую зависит от протокола маршрутизации, который должен быть правильно выбран согласно решаемой задаче мониторинга или контроля параметров внешней среды, или же протокол должен быть в некоторой степени универсален [17, 61].

*Жизненным циклом беспроводной сенсорной сети*называется интервал времени между началом функционирования и гибелью последнего из функционирующих сенсорных узлов [105].

Протоколы маршрутизации в БСС решают задачи:

1. Самоорганизация узлов сети (самоконфигурирование, самовосстановление);
2. Маршрутизация и адресация узлов;
3. Минимизация энергопотребления узлов сети и увеличение общего времени жизни всей сети;
4. Сбор и агрегация данных;
5. Скорость передачи и обработки данных в сети;
6. Максимизация зоны покрытия сети;
7. Качество обслуживания (QoS);

Протоколы маршрутизации для БСС отвечают за поддержку маршрутов в сети и должны гарантировать надежную связь даже в жестких неблагоприятных условиях. Многие протоколы маршрутизации, управления электропитанием, распространения данных, были специально разработаны для БСС, где энергосбережение является существенной проблемой, на решение которой направлен протокол [74]. Другие же были разработаны для общего применения в беспроводных сетях, но нашли свое применение и в БСС. Одна из классификаций протоколов БСС [49] представлена в табл.1.1.

Таблица 1.1. Протоколы маршрутизации БСС

№	Категория протоколов	Протоколы
1.	Основанные на местоположении узлов	MECN, SMECN, GAF, GEAR, Span, TBF, BVGF, GeRaF
2.	Направленные на агрегацию данных	SPIN, Directed Diffusion, Rumor Routing, COUGAR, ACQUIRE, EAD, Information-Directed Routing, Gradient-Based Routing, Energy-aware Routing, Information-Directed Routing, Quorum-Based Information Dissemination, Home Agent Based Information Dissemination
3.	Иерархические	LEACH, PEGASIS, HEED, TEEN, APTEEN
4.	Основанные на мобильности	SEAD, TTDD, Joint Mobility and Routing, Data MULES, Dynamic Proxy Tree-Base Data Dissemination
5.	Мульти-ориентированные	Sensor-Disjoint Multipath, Braided Multipath, N-to-1 Multipath Discovery
6.	Основанные на гетерогенности	IDSQ, CADR, CHR
7.	Основанные на качестве обслуживания (QoS)	SAR, SPEED, Energy-aware routing

1.2.1. Протоколы, основанные на местоположении узлов

Связь между узлами основана на ихместорасположении. Это может быть также применено для вычисления расстояния между двумя определенными узлами с целью оценки потребления энергии.

1.2.1.1. GeographicAdaptiveFidelity (GAF)

Энергосберегающий протокол. В основе протокола лежит принцип проецирования на виртуальную решетку (рис.1.2) местоположений сенсорных узлов, получаемых с помощью GPS или других систем. Такое представление позволяет оценить стоимость маршрутизации пакета до целевого узла, где стоимость выражается в энергозатратах на передачу пакета в соответствие с энергетической моделью [1]. Чем дальше располагается квадрант узла-адресата, тем стоимость выше. Причем, узлы, размещенные в одном и том же квадранте, будут равны по стоимости маршрутизации пакета до них.

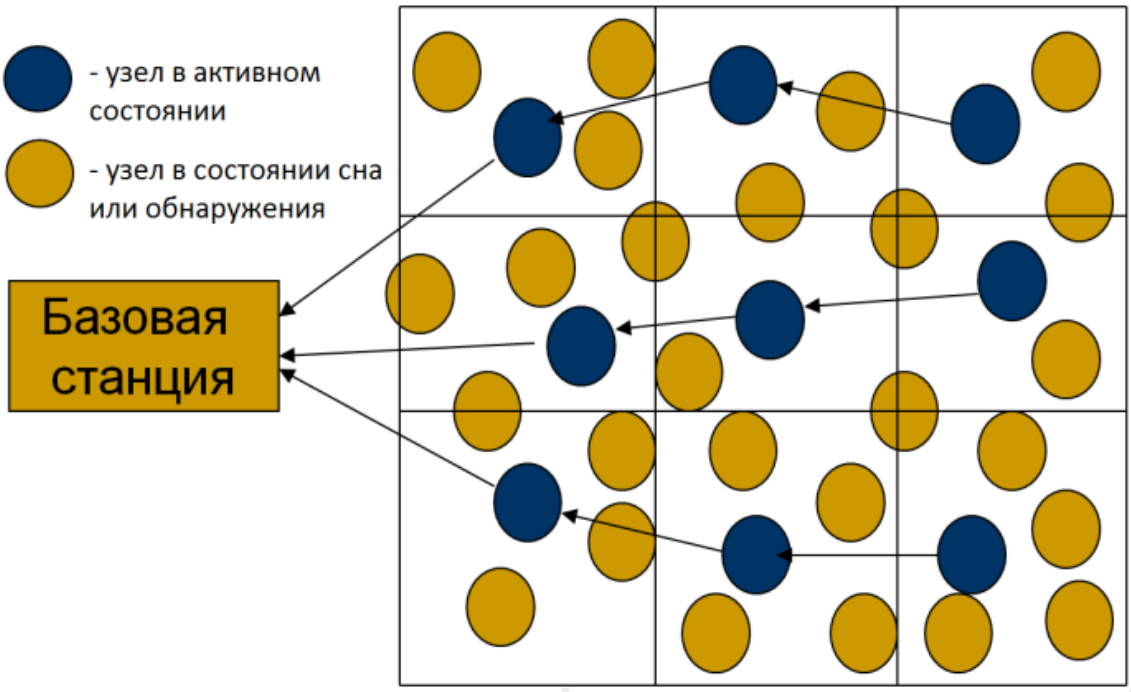


Рисунок 1.2. Пример виртуальной решетки в GAF

Площадь S_{GAF}, занимаемая одной ячейкой (квадрантом) решетки зависит от максимального радиуса радиовидимости беспроводных узлов R. Размер каждой ячейки R определяется в соответствии со следующим условием:

$$r^2 + (2r)^2 \le R^2 \quad (1.1),$$

где $r \le \frac{R}{\sqrt{5}}$ (1.2).

Следовательно, максимальная площадь ячейки определяется как:

$$S_{GAF} = (\max r)^2 = \frac{R^2}{5} \qquad (1.3)$$

Каждый из узлов сети может находиться в трех состояниях: обнаружение, активное сон (рис.1.3):

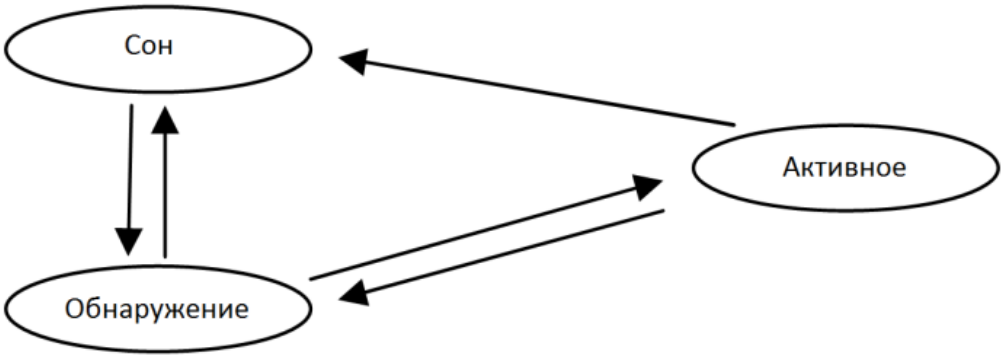


Рисунок 1.3. Диаграмма перехода состояний GAF

1.2.1.2.GeographicandEnergy-AwareRouting (GEAR)

Маршрутизация основана на знании каждым узлом своего местоположения узлов с помощью GPS (или другим систем) и об уровне своей остаточной энергии.

Стоимость передачи пакета данных до каждого соседнего узла N_i рассчитывается как:

$$C(N_i, R) = \alpha d(N_i, R) + (1-\alpha) e(N_i) \qquad (1.3),$$

где α - настраиваемый вес, $d(N_i, R)$ - нормализованное расстояние от N_i до области с центром R, e - нормализованная потребляемая энергия в N_i.

После осуществления каждого следующего хопа меняется стоимость передачи пакета. Когда следующий хоп $N_{\min}$ выбран, стоимость передачи рассчитывается следующим образом:

$$h(N, R) = h(N_{\min}, R) + C(N, N_{\min}) \qquad (1.4),$$

где $C(N,R)$ - стоимость передачи пакета от N до $N_{\min}$.

GEARиспользует рекурсивный алгоритм географической эстафетной передачи для распространения пакета внутри целевого региона [59].

1.2.1.3.Trajectory-Based Forwarding (TBF)

В TBF [40] узел-источник определяет маршрут в пакете, но явно не указывает маршрут в виде прыжков "хопов" (рис. 1.4).

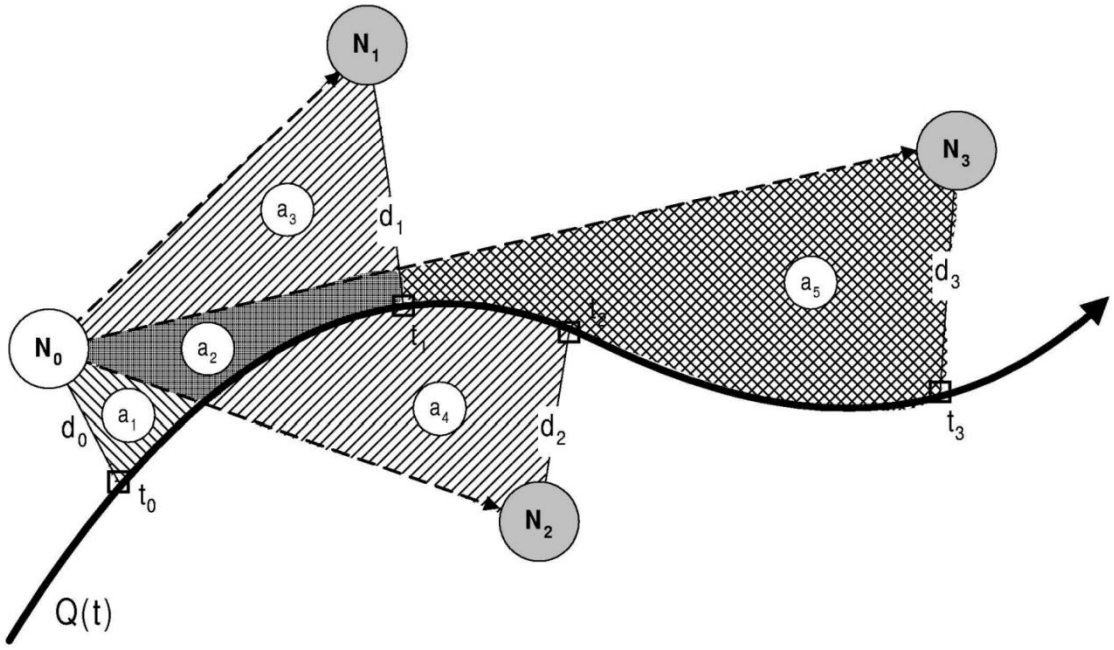


Рисунок 1.4. Задание маршрута в пакете TBF

Основываясь на информации о местоположении своих соседей, узел-ретранслятор принимает решение определить следующий «хоп», который будет являться самым приближенным к маршруту, установленному сенсором-источником. При этом,с каждым хопом уменьшается радиус области A_i между узлами N_0 и N_i:

$$R_i = \frac{A_i}{t_i - t_0} = \frac{Area(N_0, N_i, Q(t_0), Q(t_i))}{t_i - t_0} \qquad (1.5)$$

где t_0 и t_i - опорные точки маршрута Q – полином, описывающий функцию маршрута.

1.2.1.4. Bounded Voronoi Greedy Forwarding [BVGF]

На основании местоположения сенсоров строится диаграмма Вороного [3]. Маршрутизация производится в соответствии с этой диаграммой, при этом, для определения пути вычисляется минимальное Евклидово расстояние до пункта назначения среди всех имеющих права соседей. BVGF не рассматривает энергию как метрику маршрута [53, 86].

1.2.1.5. GeographicRandomForwarding (GeRaF)

ПредложенЗорзииРао [60]. Концепция протокола состоит в передаче пакета от источника к приемнику с помощью ретрансляции пакетов первого, зона покрытия которого называется зоной передачи. В свою очередь, эта зона разделена на несколько областей со своими приоритетами N_p. Каждая область включает все узлы, где расстояние до приемника определяется как:

$$D-1+\frac{(i-1)}{N_p} \le \gamma < D-1+\frac{i}{N_p},\ i=1,2,...,N_p \quad (1.6),$$

где D – расстояние от передающего узла до приемника, i– номер области зоны передачи, γ – остаток расстояния после первого хопа. При этом:

$$D-1 \le \gamma \le D \quad (1.7).$$

Самой приоритетной является область, самая близкая к приемнику. Источник стремится выбирать ретранслятор в самой высокоприоритетной области, так, чтобы за наименьшее количество «хопов» передать пакет приемнику. Если такого ретранслятора нет, то выбирается область с более низким приоритетом. В случае отсутствия сенсора-ретранслятора, по достижении определенного количества попыток, пакет будет отброшен.

1.2.1.6. Minimum Energy Communication Network (MECN)

Вычисляется остовое дерево с корнем от приемника, которое называется минимальной мощностной топологией, содержащей только

минимальные пути, на основании количества остаточной энергии, от каждого сенсора до приемника:

$$C(r)=\sum_{i=1}^{k-1}\left(p(u_i,u_{i+1})+c\right) \quad (1.8),$$

$$r=(u,u_1,...,v) \quad (1.9),$$

где r - путь между u и v, который охватывает $k-1$ промежуточных узлов $u_1,...u_{k-1}$.

Концепция основана на расположении сенсоров на плоскости и состоит из двух главных фаз, а именно: построение графа включения и распределения стоимостей пути [44].

1.2.1.7. Small Minimum-Energy Communication Network (SMECN)

Протокол SMECN [29] был предложен для улучшения MECN. Каждый сенсор производит обнаружение своих соседей, с помощью широковещательного сообщения. В начале работы используется некоторый уровень начальной энергии p, с мощностью которого рассылается сообщение. Если не один из соседей не ответил, то эта энергия увеличивается и сообщение рассылается заново. В SMECN также строится граф и вычисляется самый короткий по стоимости энергозатрат путь, как и в MECN.

1.2.2. Протоколы, направленные на агрегацию данных

В протоколах, направленных на агрегацию данных, сенсоры, располагающиеся между источниками и БС, могут осуществлять агрегацию данных и посылать БС уже сведенные данные. Этот процесс позволяет сенсорным узлам экономить энергию.

1.2.2.1. SensorProtocolsforInformationviaNegotiation (SPIN-1, -2)

Протоколы SPIN [18, 27] основаны на двух ключевых механизмах: на согласовании и адаптации ресурса. SPIN позволяет сенсорам производить

согласование друг с другом перед любым распространением данных в сети, во избежание введения бесполезной и избыточной информации в сеть. SPIN использует мета-данные, как описатели данных, которые сенсоры распространяют. Понятие мета-данных предотвращает возникновение наложения для данного сенсора [50].

В семействе SPIN существует два протокола: SPIN-1 (или SPIN-PP) и SPIN-2 (или SPIN-EC) [27]. В то время как SPIN-1 использует механизм согласования перед любым распространением данных в сети, чтобы уменьшить потребление ресурсов сенсорами, а также во избежание введения бесполезной и избыточной информации в сеть. SPIN-2 использует ресурсо-уведомляющий механизм для энергосбережения.

1.2.2.2. Directed Diffusion

Протокол [23, 24] имеет несколько основных компонентов: именование данных, интересы и градиенты, распространение данных и укрепление. Процесс передачи данных в таком протоколе описывается как *направленная диффузия*. В начале направленной диффузии приемник определяет низкую скорость передачи данных для всех поступающих событий. После этого приемник может "укрепить" один определенный сенсор, позволив ему увеличить скорость передачи, отправляя "сообщение-интерес" приемнику. Аналогично, если соседний сенсор получит это "сообщение-интерес" и обнаружит, что у "интереса" отправителя есть более высокая скорость передачи данных чем прежде, и эта скорость передачи данных выше чем любого существующего градиента, то это "укрепит" один или более его сенсоров-соседей.

1.2.2.3. Rumor Routing

Ключевым механизмом протокола является агент – пакет с большим временем жизни, который пересекает сеть и сообщает каждому сенсору о событиях, которые он встретил на своем пути во время пересечения сети.

Агент будет путешествовать по сети до определенного числа "хопов" и затем прекратит свою жизнь. Каждый сенсор, включая агента, содержит список событий, в котором приведены пары событие-расстояние, где расстояние обозначает фактическое расстояние, выраженное в количестве "хопов" к соответствующему событию от сенсора, который он посетил. Когда агент встречается с сенсорами на своем пути, то он синхронизирует список событий в каждом сенсоре, так, чтобы в них были кратчайшие пути по отношению к событиям, происходящим в сети [7].

1.2.2.4. Cougar

В Cougar [57] сеть представляется в виде огромной распределенной базы данных, где каждый сенсор имеет своё подмножество данных. Протоколом предоставляются пользовательские и прикладные программы с возможностью декларативных запросов данных, которые детектируют сенсоры-источники. Такой подход позволяет абстрагировать пользователя от способа выполнения запросов и обработки данных, предоставляя в конечном итоге только результат. Протокол выделяет *уровень запросов*, который находится между сетевым и прикладным уровнем и связан со своим прокси-запросом. Прокси-запрос предоставляет высокоуровневые сервисы через запросы, которые могут быть получены от узла-шлюза. Cougar более выгоден, если набор детектируемых данных мог бы быть синтезирован и представлен в виде одного объекта, понятного для пользователя [107].

1.2.2.5. Active Query Forwarding in Sensor Networks (ACQUIRE)

Протокол ACQUIRE [45] предоставляет механизм запросов для именованных данных. Это обеспечивает оптимизацию для ответа на определенные типы запросов, которые называются в протоколе как «*одно-сложные запросы для реплицируемых данных»*. Запрос ACQUIRE состоит из нескольких подзапросов, для которых несколько простых ответов обеспечиваются соответствующими им сенсорами. Каждому подзапросу

приходит ответ на основании хранимых в текущий момент данных в соответствующем сенсоре. ACQUIRE позволяет сенсору вводить в сеть активный запрос, либо случайно, либо по указанной траектории до тех пор, пока на запрос не ответят сенсоры, используя локализованный механизм обновления. В отличие от других механизмов запросов, ACQUIRE, позволяет вводить сложный запрос в сеть, который при этом, будет отправлен пошагово через последовательность сенсоров.

1.2.2.6. Energy-Aware Data-Centric Routing (EAD)

EAD [6] выстраивает дерево радиопередачи с корнем в роли шлюза, активных сенсоров в роли ствола и сенсоров с выключенными радио-модулями, которые представляют листья. Протокол аппроксимирует оптимальное дерево охвата сенсоров с минимальным числом листьев, уменьшая размер ствола, сформированной активными сенсорами. Подход EAD позволяет информировать о количестве энергии и помогает увеличить жизненный цикл сети. Шлюз играет роль приемника данных или приемника событий, тогда как каждый сенсор действует как источник событий или хранилище данных.

1.2.3. Иерархические протоколы

Для работы иерархических протоколов маршрутизации необходимо всё множество сенсоров разделить на кластеры (группы узлов) – рис. 1.5. Каждым кластером управляет специальный узел, называемый головным кластерным узлом (ГКУ), который отвечает за координирование передачи и маршрутизацию детектируемых данных в своем кластере, а также до базовых станций. Объединение сетевых узлов в кластеры позволяет продлить эффективное время работы БСС [25, 85].

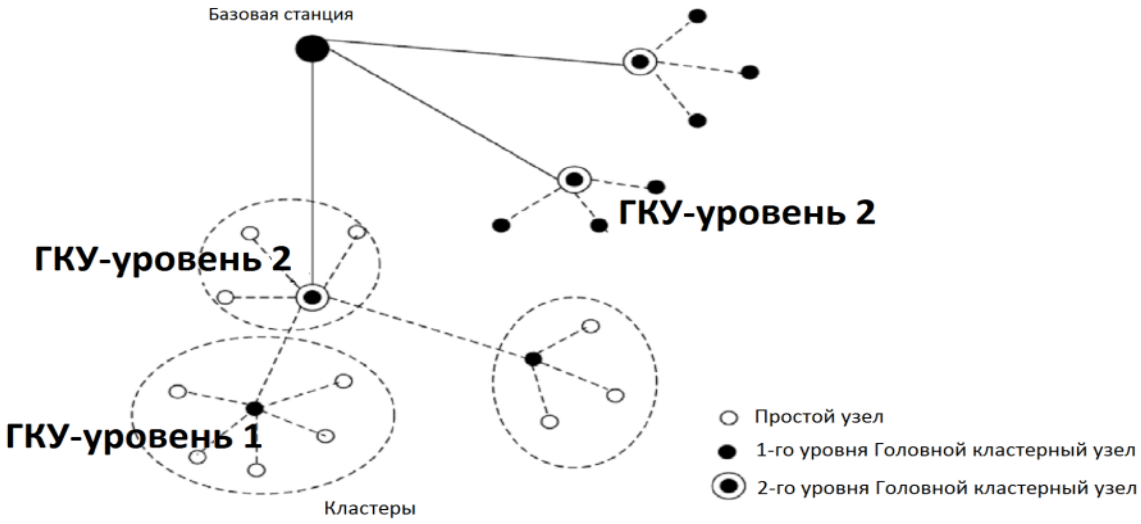


Рисунок 1.5. Основанная на кластерах иерархическая модель

1.2.3.1. Low-energyadaptiveclusteringhierarchy (LEACH).

В начале работы LEACH [19, 21, 42],узлы самоорганизуютсяв кластеры посредством выбора головных кластерных узлов (ГКУ), при этом каждый узел предлагает себя в качестве ГКУ с определённой вероятностью: каждый узел генерирует случайное число $\beta \in [0;1]$. Если $\beta < T(n,b)$, то узел становится ГКУ на текущий раунд.

$$T(n,b) = \begin{cases} \dfrac{\mathrm{b}}{1-\mathrm{b}\left(\mathrm{r}\cdot \mathrm{mod}\left(\dfrac{1}{\mathrm{b}}\right)\right)}, \text{если } n \in \mathrm{G} \\ \text{иначе } 0 \end{cases} \qquad (1.10),$$

где b– желаемый процент становления ГКУдля данного узла, r–текущий раунд, G – множество узлов, которые не стали ГКУ в последних $\dfrac{1}{b}$ раундах.

После выбора ГКУ все узлы начинают передавать детектируемые данные своему ГКУ. Таким образом, образуются кластеры во главе с узлами, которым передаются все детектируемые из внешней среды данные. Каждый ГКУ принимает данные, производит их обработку и отправляет на базовую станцию (БС).Периодически в LEACH происходит перевыбор ГКУ на

основании случайной выборкивысокоэнергоэффективных узлов. В результате происходит перекластеризация, что необходимо для распределения энергетической нагрузки по сети[8, 112].

1.2.3.2. Power-Efficient Gathering in Sensor Information Systems (PEGASIS).

Протокол PEGASIS [20, 22, 30] является расширением протокола LEACH, который формирует цепи (рис. 1.6) из сенсорных узлов вместо кластеров в LEACHтак, чтобы каждый узел передавал и получал от соседа. При этом только один узел избирается из цепи для осуществления передачи данных на базовую станцию (БС, приемник). Данные агрегируются и перемещаются от узла к узлу, соединяясь, и в конечном счете, попадая на БС.

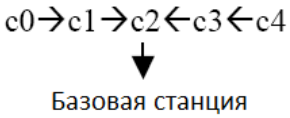


Рисунок1.6. МаршрутизациявPEGASIS

1.2.3.3. Hybrid, Energy-EfficientDistributedClustering (HEED).

Протокол HEED [58] расширяет базовую схему протокола LEACH, используя остаточную энергию и уровень узла или же плотность размещения узлов в качестве метрики для выбора кластера с целью достижения баланса энергии в кластерах. В соответствии с алгоритмом в HEED периодически происходит выбор головных кластерных узлов согласно комбинации двух параметров кластеризации. Основной параметр $E_{residual}$- остаточная энергия каждого сенсорного узла (используется в вычислении вероятности присвоения ему статуса ГКУ), и второй параметр C_{prop}- внутрикластерная коммуникационная стоимость как функция плотности кластера или же используется уровень узла (то есть число соседей):

$$CH_{prop} = C_{prop} + \frac{E_{residual}}{E_{\max}} \qquad (1.11),$$

где $E_{\max}$ - начальная энергия узла после включения.

$E_{residual}$используется для того, чтобы вероятностно выбрать начальный набор головных кластерных узлов, в то время как C_{prop}используется для того, чтобы разорвать связи. Кластеризация HEED увеличивает жизненный циклсети больше, чем LEACH, поскольку последний, беспорядочно выбирает головные кластерные узлы (и, следовательно, размер кластера), что может привести к более быстрому выходу из строя некоторых узлов.

1.2.3.4. Threshold Sensitive Energy Efficient Sensor Network Protocol (TEEN).

Протокол TEEN [37] является иерархическим протоколом кластеризации, который группирует сенсорные узлы в кластеры с выбором соответствующего ГКУ. При этом используется несколько иерархических уровней кластеров со своим ГКУ соответственно, каждый из которых агрегирует данные и передает ГКУ более высшего уровня. На самом высоком уровне ГКУ передают данные БС.

К важным особенностям TEEN относится его пригодность для работы критичных ко времени выполнениязадач сбора данных. Кроме того, так как передача сообщения расходует больше энергии, чем детектирование данных, то потребление энергии а такой схеме меньше, чем в проактивных сетях. Также стоит отметить, что, TEEN не подходит для сбора данных, где существует необходимость в частой передаче сообщений, так как пользователь может вообще не получить данные, если пороги срабатывания не достигнуты.

1.2.3.5. Adaptive Periodic Threshold Sensitive Energy Efficient Sensor Network Protocol (APTEEN).

Протокол APTEEN– гибридный, основанный на кластеризации протокол маршрутизации, являющийся продвинутой версией TEEN [38]. В протоколе сенсорные узлы периодически передают свои данные и реагируют на любое внезапное изменение значения вариативного параметра, сообщая о соответствующих значениях своим ГКУ. Архитектура APTEEN является такой же как и в TEEN, используя концепцию иерархической кластеризации для обеспечения энергоэффективной связи между сенсорами-источниками и БС. APTEEN поддерживает три различных типа запросов для получения данных от узлов БСС:исторический запрос для анализа предыдущих значений данных; одноразовый запрос для получения снимка представления сети;постоянные запросы для мониторинга событий в течение заданного промежутка времени

APTEEN гарантирует более низкое энергопотребление, чем TEEN и большее число действующих сенсорных узлов [33].

1.2.4. Протоколы, основанные на мобильности

Основанные на мобильности протоколы предполагают мобильность приемника. Для данной задачи основным является требование гарантированной доставки данных, порожденных сенсорами-источниками,мобильной БС.

1.2.4.1. Joint Mobility and Routing Protocol.

В данном протоколе [49], сенсорные узлы, окружающие мобильную БС, в результате её движения, меняют свое местоположение относительно неё в течение времени. При этом каждый сенсорный узел периодически может работать в качестве ретранслятора данных, направляемых к БС. Это

позволяет осуществлять балансировку нагрузки маршрутизации данных на всех сенсорных узлах.

В протоколе существует несколько стратегий движения БС, при этом, предполагается, что поле из сенсорных узлов - круг: движение по концентрическим окружностям; перемещение в окружностях; симметричная стратегия (движение по границе сети). Наиболее оптимальной в большинстве случаев считается последняя стратегия.

1.2.4.2. ScalableEnergy-EfficientAsynchronousDissemination (SEAD):

В протоколе [26] распространение данных от сенсоров-источников может осуществляться до различных БС. Протокол состоит из трех главных компонентов: построение дерева распространения (d-дерево); распространение данных; поддержание связей с мобильными БС.

Предполагается, что сенсоры знают о своих собственных географических местоположениях. Каждый сенсор-источник строит свое дерево распространения данных, где корнем является он сам. Все деревья распространения для всех сенсоров-источников строятся отдельно.

1.2.4.3. DynamicProxyTree-BasedDataDissemination.

Протокол [9] строит дерево для каждого сенсора-источника, соединяющее его с несколькими мобильными БС. При этом считается, что сами сенсоры-источники стационарны, но целевые объекты детектирования являются мобильными. Вследствие мобильности целевого объекта может измениться сенсор-источник, с которого БС получает данные. Это происходит при достижении величины расстояния некоторого порога. Следовательно, ближайший к целевому объекту сенсорный узел может стать источником. Каждый источник является прокси-источником. Аналогичное верно для БС (приемников). Источник и прокси-приемники являются временными в том смысле, что они сменяются по мере того, как меняются источники в зависимости от положения целевого объекта, а также в

зависимости от того, как перемещаются приемники. Прокси уменьшают стоимость передачи и запроса данных у источника и прокси-приемников [10].

1.2.5. Мульти-ориентированные (многопутевые) протоколы

При рассмотрении передачи данных между сенсорами-источниками и приемниками, имеют место быть две парадигмы маршрутизации: однопутевая маршрутизация и многопутевая маршрутизация. В однопутевой маршрутизации, каждый сенсор-источник посылает свои данные к приемнику через кратчайший путь. В многопутевой маршрутизации, каждый сенсор-источник находит первые k кратчайших путей к приемнику и делит его нагрузку равномерно среди этих путей [11].

1.2.5.1. DisjointPaths.

Является [12, 31] многопутевым протоколом маршрутизации несвязанных сенсоров, где вычисляется несколько альтернативных путей передачи данных, у которых нет ни одного общего с основным маршрутом сенсорного узла. В маршрутизации несвязанных сенсоров, основной путь является наилучшим из имеющихся, тогда как альтернативные пути менее желательны, поскольку у них есть более длительное время ожидания. Несвязность делает эти альтернативные пути независимыми от основного. Таким образом, если отказ происходит на основном пути, то это остается локальной проблемой и не затрагивает ни один из альтернативных путей. Приемник может определить, кто из соседних сенсорных узлов может ему предоставить данные наивысшего качества, где последнее характеризуется низкими потерями или самой малой задержкой сетевого флудинга. Хотя несвязные пути более гибки к отказам сенсоров, они могут быть потенциально более длинными, чем основной путь и, таким образом, менее энергосберегающими.

1.2.5.2. BraidedPaths.

Является протоколом маршрутизации [12, 32] со множеством «плетеных» путей, являясь в свою очередь версией протокола Disjoint Paths с более ослабленными правилами построения альтернативных путей. Отличие заключается в том (рис. 1.7), что альтернативные пути могут включать в себя некоторые узлы основного пути. При этом, перед построением альтернативных путей, сначала должен быть вычислен основной маршрут. Таким образом, такие альтернативные пути называются частично несвязными путями.

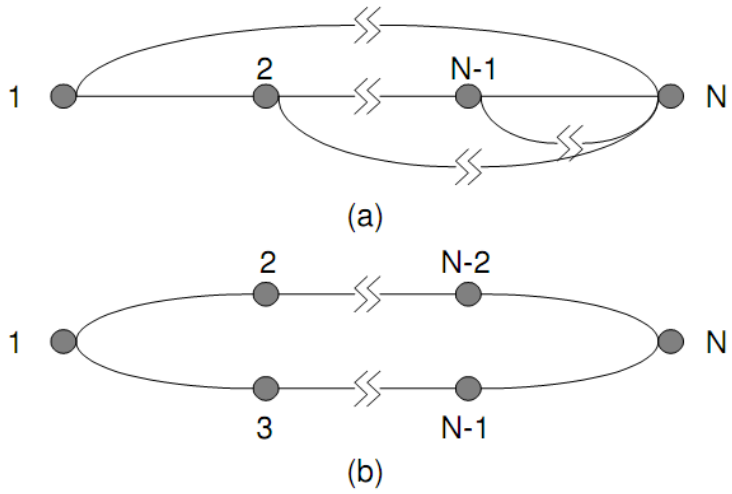


Рисунок 1.7.Сравнение путей протоколов DisjointPaths (a) и BraidedPaths (b)

1.2.6. Основанные на гетерогенности протоколы

В архитектуре гетерогенной сенсорной сети существует два типа сенсорных узлов: сенсоры с линией питания, которые не имеют никакого ограничения энергии; сенсоры с питанием от батареи, имеющие ограничение времени работы и, следовательно, должны использовать свой доступный энергоресурс экономно, минимизируя затраты на вычисления и передачу данных.

1.2.6.1. Information-Driven Sensor Query (IDSQ).

В протоколе [12, 31] для сохранения энергии, необходимо, чтобы в активном состоянии находилось только некоторое подмножество сенсоров, которые вводиться в строй по мере наличия в разных частях сети определенных событий. Выбор подмножества активных сенсоров, у которых есть наиболее полезная информация, сбалансирован коммуникационной стоимостью, необходимой между этими сенсорами. Полезная информация может быть найдена на основе аппроксимации времени и пространства, - где и когда могут произойти события. В протоколе IDSQ первым шагом должен быть выбран один сенсор в качестве ведущего в кластере сенсоров. Этот ведущий будет ответственен за оптимальный выбор сенсоров на основании некоторой меры служебной информации.

1.2.6.2. Cluster-Head Relay Routing (CHR).

Протокол предполагает формирование гетерогенной сети посредством использования двух категорий сенсоров: большого количества низкокачественных и небольшого высококачественных. Первые обозначаются как L, а последние как H. Все сенсоры имеют информацию о своих местоположениях. Протокол делит сеть на кластеры, которые формируются с помощью L-сенсоров, во главе которых ставится один из числа H-сенсоров.L-сенсоры детектируют данные и передают между собой с помощью «мультихопов» на малые расстояния до своего главного сенсора в рамках данного кластера, который выбирается из числа H-сенсоров. Последние передают данные на дальние расстояния до других H-сенсоров, на пути к приемнику [14, 5].

1.3. Исследование топологий, моделей связности узлов и выявление наиболее эффективных

Существует несколько моделей связности узлов в беспроводных сенсорных сетях, которые определяют порядок следования пакета при

доставке его от узла к БС. Среди этих моделей наиболее известными являются:

1. Одноинтервальные модели.
    1. плоские;
    2. иерархические.
2. Многоинтервальные модели.
    1. плоские;
    2. иерархические.

В одноинтервальных моделях (рис. 1.8.1-1.8.2) все сенсорные узлы передают свои данные напрямую на БС. При этом, части узлов может не хватить мощности приемопередатчика для достижения БС. В таком случае эти сенсорные узлы будут не задействованы в работе сети.

Рисунок 1.8.1 Одноинтервальная плоская модель

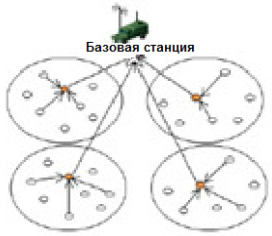


Рисунок 1.8.2 Одноинтервальная иерархическая модель

Такие модели можно применять только при размещении каждого узлов на расстоянии, достаточном для достижения БС. В противном случае узлы просто не будут включены в состав сети.

Эти модели являются неэффективными [51] при использовании крупных сетей, поскольку затраты энергии на передачу данных становятся большими и в худшем случае, БС может быть недосягаема для ряда узлов. Сети, построенные на основании данной топологии, являются не масштабируемыми, а на размещение узлов в целевой области накладывается ряд ограничений.

В многоинтервальных моделях (рис. 1.8.3 –1.8.4) данные от каждого сенсорного узла передаются до БС опосредованно. В многоинтервальной

плоской модели, вследствие того, что все узлы должны хранить одну и ту же информацию, например, такую как таблица маршрутизации, накладные расходы на потребление энергии могут возрасти. С другой стороны, в многоинтервальной иерархической модели, сенсорные узлы обеспечивают низкие накладные расходы и потребление энергии, поскольку отдельные ГКУ агрегируют данные и передают их на БС [69, 70]. Кроме того, в многоинтервальной плоской модели, беспроводная среда разделяется и управляются отдельными узлами, что приводит к низкой эффективности использования ресурсов. В многоинтервальной иерархической модели, ресурсы могут быть выделены ортогонально каждому кластеру для уменьшения коллизий между кластерами и повторного их использования [2, 51].

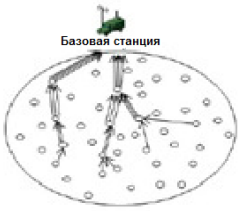


Рисунок 1.8.3 Многоинтервальная плоская модель

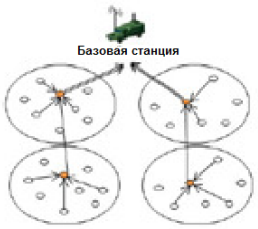


Рисунок 1.8.4 Многоинтервальная иерархическая модель

1.3.1.Связность в иерархических протоколах маршрутизации

В иерархических протоколах (протоколах, основанных на кластеризации), сеть делится на кластеры (группы узлов). Это позволяет эффективнее использовать энергетические ресурсы сети, тем самым продлевая её жизненный цикл [2, 51].

Разделение сети на кластеры может производиться на основании определенных критериев, например таких как:

- географическое местоположение узлов (те узлы, которые расположены рядом друг с другом, то есть являются соседями, попадают в один и тот же кластер);

- уровень остаточной энергии;
- мощность сигнала (узлы попадают в один и тот же кластер на основании досягаемости друг друга по мощности сигнала);
- другие критерии.

Во главе каждого кластера $K_i \in \{K_i \mid i = 1,...L\}$ выбирается ГКУ - $CH_i \in K_i$, на который передаются данные с других узлов $q \in K_i$ в рамках данного кластера K_i. ГКУ собирает данные со своего кластера и передает их дальше по сети. При этом ГКУ может оптимизировать данные, производя операции сжатия и фильтрации.

В иерархических протоколах отдельное место занимает понятие связности. Связность делится на две категории:

1. Внутрикластерная (intra-cluster);
2. Межкластерная (inter-cluster).

Внутрикластерная связность обеспечивает передачу данных между узлами внутри кластера, а межкластерная - между другими кластерами, а также БС [46].

1.3.1.1. Внутрикластерная связность

Внутрикластерная связность может быть одноинтервальной (one-hop) или многоинтервальной (multi-hop).

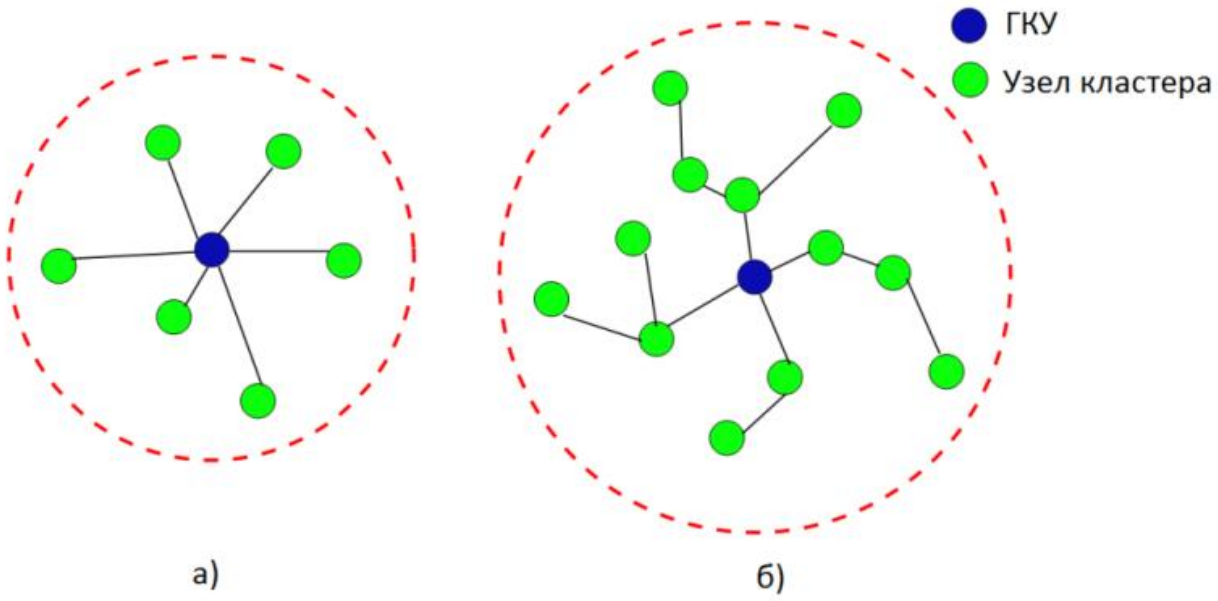


Рисунок 1.9. а) Одноинтервальная внутрикластерная связность,
б) Многоинтервальная внутрикластерная связность

При одноинтервальной связности пакеты с данными $data$ передаются напрямую к ГКУ $q_i \to data \to CH$, а при многоинтервальной – опосредованно, через другие узлы $q_i \to data \to q_{i+1} \to data \to ... \to data \to$ $\to CH$.

1.3.1.2. Межкластерная связность

Межкластерная связность также как и внутрикластерная может быть одноинтервальной (one-hop) или многоинтервальной (multi-hop).

При одноинтервальной связности, данные, агрегированные ГКУ передаются напрямую на базовую станцию (рис. 1.10). Данный подход имеет существенный недостаток – ГКУ не всегда смогут передавать данные напрямую базовой станции, в виду их удаленности от неё. Чем дальше будет располагаться ГКУ, тем слабее будет сигнал или же БС будет вообще вне зоны радиуса действия приемопередатчика ГКУ. Это существенное ограничение снижает как возможности использования протоколов с одноинтервальной межкластерной связностью, так и их масштабируемость.

Многоинтервальная связность позволяет каждому ГКУ, вне зависимости от его удаленности от БС, передавать ей данные. Пакеты с данными передаются опосредованно – через другие ГКУ (рис 1.11). Но здесь возникает другая проблема: ГКУ может не хватить мощности, чтобы связаться друг с другом. Денная проблема межкластерной связности может возникнуть в виду особенностей деления на кластеры. Например, радиус действия ГКУ4 охватывает ГКУ3, поэтому возможна опосредованная передача: ГКУ4->ГКУ3->БС. ГКУ1 и ГКУ2 не достижимы для остальных ГКУ [1].

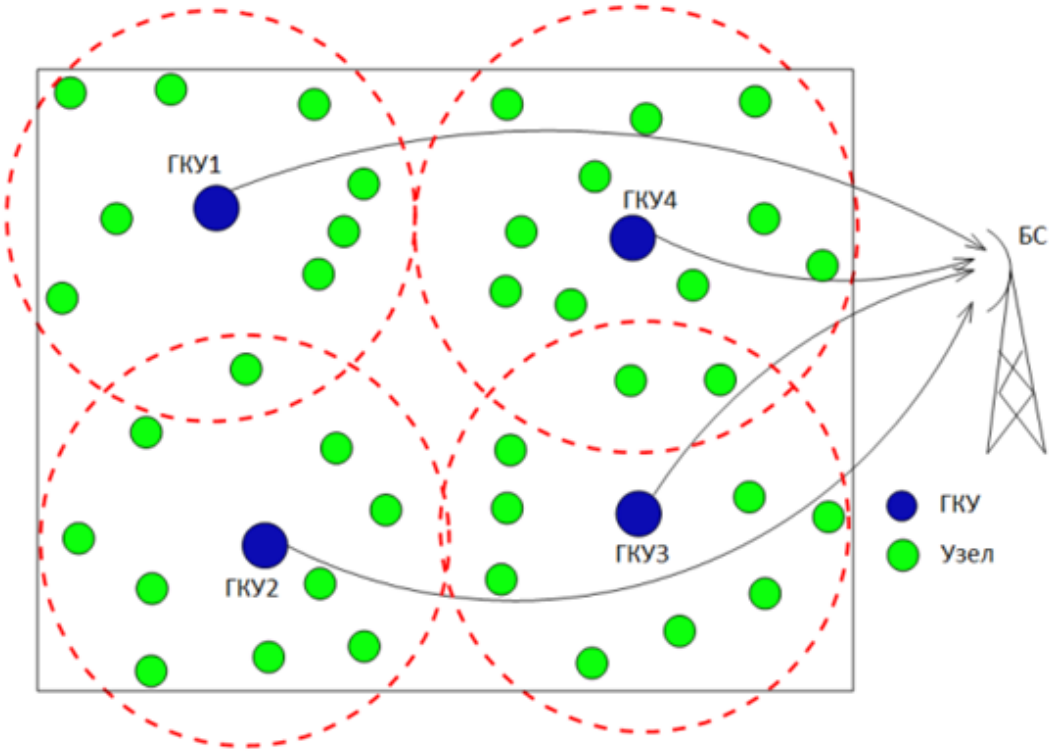


Рисунок 1.10. Одноинтервальная межкластерная связность

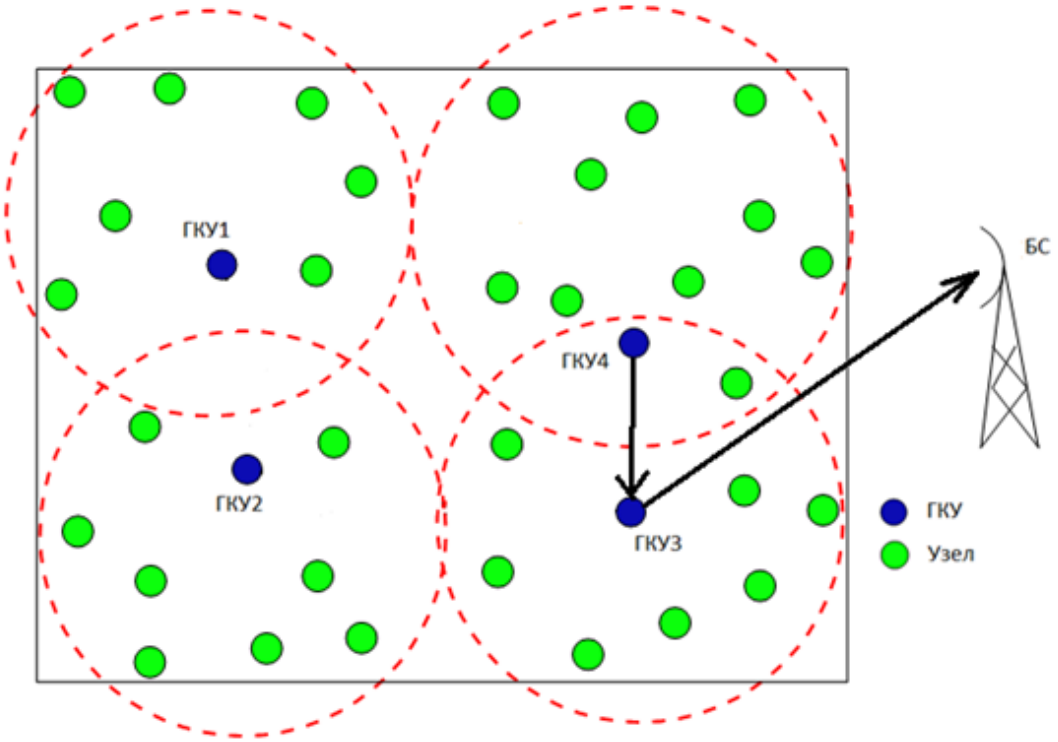


Рисунок 1.11. Многоинтервальная межкластерная связность

Решением этой проблемы будет использование релеев – шлюзов, через которые данные между двумя ГКУ будут передаваться опосредованно. Такие шлюзы могут быть двух категорий: распределенные и общие.

Общими называются такие шлюзы, которые располагаются между радиусами передачи 2-х ГКУ и реализуют двух-интервальную передачу: ГКУ1 -> Общий шлюз -> ГКУ2. Когда 2 ГКУ не имеют Общего шлюза, то они могут связаться друг с другом через 3 интервала (3 хопа) с помощью Распределенных шлюзов, каждый из которых расположен в своём кластере со своим ГКУ. Каждый из Распределенных шлюзов доступен только для своего ГКУ в рамках кластера и Распределенного шлюза в другом кластере, где располагается 2-ой ГКУ. Таким образом, Распределенными называются такие шлюзы, которые реализуют трех-интервальную передачу: ГКУ1 -> Распределенный шлюз -> Распределенный шлюз2 -> ГКУ2 [43].

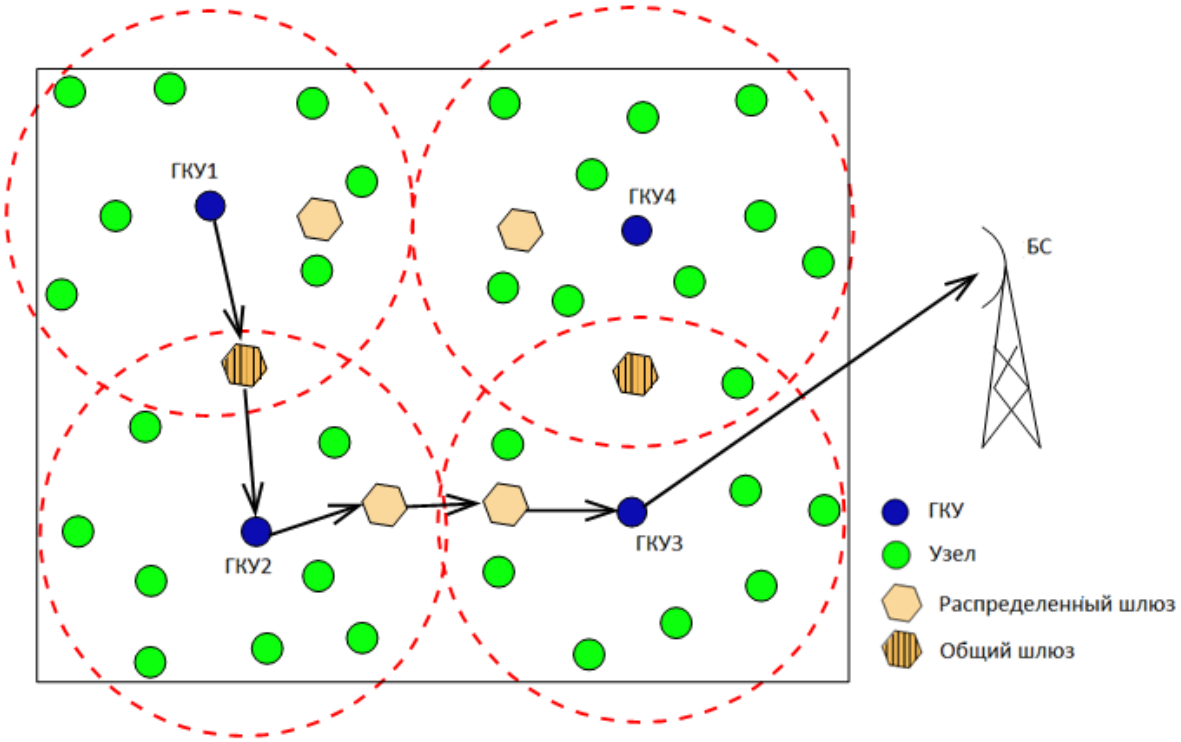


Рисунок1.12. Многоинтервальная межкластерная связность с релеями

1.3.2. Иерархическая структура

Межкластерная и внутрикластерная связности, объединяясь, образуют иерархию, что соответствует названию данной группы протоколов. Узлы, подчиненные ГКУ в рамках кластера верхнего уровня могут быть субкластерами (кластерами нижнего уровня), которые в свою очередь могут иметь свои субкластеры с соответствующими ГКУ [104]. Таким образом, субкластеры будут представлять собой внутрикластерную связность отдельного ГКУ и, при этом, иметь свою межкластерную и внутрикластерную связности [34].

Выводы по главе 1

В данной главе были рассмотрены протоколы по одной из известных классификаций [49] для БСС.

Протоколы, основанные на местоположении узлов, подходят для использования на локациях, где не будет проблем связи со спутниками GPS/ГЛОНАС [71, 72]. В большей степени такие локации - это открытые

пространства, где возможна беспрепятственная связь со спутниками. Также одним из важных условий для корректной работы БСС под управлением таких протоколов являются погодные условия. В целом, факт зависимости от систем определения местоположения является самой главной проблемой данной категории протоколов, что делает их ненадежными и ставит под угрозу возможность маршрутизации данных между узлами.

Протоколы, направленные на агрегацию данных могут быть применены для работы в условиях с плохой проходимостью радиосигнала. Протоколы этой категории позволяют снизить объем маршрутизируемых данных и могут передать могут передать необходимый объем данных в значительно малых по объему пакетах.

Иерархические протоколы маршрутизации эффективны в отношении энергосбережения за счет агрегации и слияния данных [2, 51]. При передаче пакетов данных основная нагрузка приходится на ГКУ, которые получают пакеты с ведомых узлов вместо использования распространенной плоской модели передачи данных от узла А к узлу Б, где пакет на протяжении всего пути проходит множество посредников. Это позволяет уменьшить количество передаваемых сообщений БС. Иерархическая маршрутизация является двуслойной. Первый слой используется для передачи данных с ведомых узлов на ГКУ, а второй – для передачи на данных от ГКУ на БС.

Основанные на мобильности протоколы позволяют использовать в качестве БС мобильные устройства, такие как, например, планшет, смартфон, ноутбук, мобильная техника и др. Такие протоколы обеспечивают гарантированную доставку данных до БС, последняя при этом может периодически выходить из зоны досягаемости БСС.

Мульти-ориентированные (многопутевые) протоколы являются альтернативой однопутевой маршрутизации. Принцип работы протоколов состоит в выборе нескольких путей для доставки данных от узлов к БС. Вследствие данной особенности многопутевой маршрутизации -

использовании избыточных путей, в значительной степени решаются проблемы, присущие однопутевой маршрутизации - это надежность, безопасность и балансировка нагрузки на один маршрут.

Основанные на гетерогенности протоколы эффективны при использовании в сети двух типов узлов – с неограниченным источником питания и автономных.

В результате произведенного обзора протоколов сделаны следующие выводы.

Одной из главных проблем в разработке протоколов маршрутизации для БСС является эффективность использования энергии, вследствие ограниченных энергетических ресурсов сенсорных узлов. Протокол должен поддерживать работоспособность сети настолько долго, насколько это возможным, тем самым, продлевая жизненный цикл сети.

Второй проблемой является обеспечение самоорганизации, при которой в работу сети будет включено как можно больше узлов [62]. При этом конечной целью является задействование всех узлов сети [63]. Эта проблема возникает, поскольку при дислокации узлов в целевую область, вследствие препятствий, возможны проблемы с обнаружением узлами друг друга и в итоге, неверной или неполной самоорганизацией сети.

Протоколы маршрутизации играют значительную роль в функционированииБСС. Благодаря им осуществляется самоорганизация узлов и доставка пакетов оптимальными маршрутами в соответствии с алгоритмами, используемымив соответствующемпротоколе. Посредством выбора эффективного протокола маршрутизации можно оптимизировать использование ресурсов сети, таких как расход энергии, использование процессорного времени, памяти и др. Следовательно, применение эффективного протокола маршрутизации позволяет максимизировать жизненный циклсети.

В данной главе также произведен анализ существующих моделей связности узлов для БСС.

На основании данных исследований [48] многоинтервальная иерархическая модель является для БСС наиболее эффективной, так как за счет иерархической структуры с выделением ГКУ, она позволяет расходовать ресурсы автономных источников энергии каждого узла более экономно в отличие от плоских моделей. В отличие от одноинтервальных моделей, данная модель связности предоставляет возможности развертывания сети в удаленных широкомасштабных областях, что говорит о высокой масштабируемости сети [94].

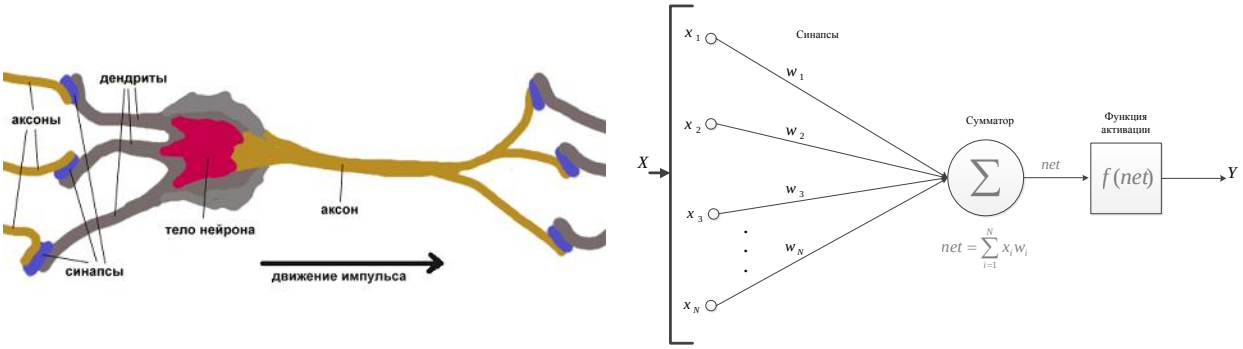
Многоинтервальная модель связности определяет эффективную топологию, где сеть поделена на отдельные кластеры, во главе каждого из которых выбирается ГКУ [35, 51]. Данные в сети передаются от одного ГКУ до БС через цепь других ГКУ опосредованно. Без наличия других ГКУ в зоне радиовидимости необходимо использовать релеи[94].

ГЛАВА 2. Исследование возможности применения нейросетевых технологий в беспроводных сенсорных сетях

2.1. Аспекты применения искусственных нейронных сетей в БСС

Искусственные нейронная сеть (ИНС) является моделью биологической нервной системы, и представляет сеть из взаимосвязанных между собой узлов, называемых нейронами с функциями активации. Всевозможные типы нейронных сетей получаются путём варьирования функцией активации и весами связей между нейронами [39].

Таким образом, биологический нейрон фактически является аналоговым представлением (рисунок 2.1а), а искусственный нейрон – его дискретной моделью[77]. Дискретная модель нейрона с входами, выходами и другими параметрами, представлена на рисунке 2.1б.



а) аналоговый нейрон б) дискретный нейрон (персептрон)

Рисунок 2.1 Аналоговый и дискретный нейроны.

Принцип работы нейрона можно описать следующим образом. На входы $x_1, x_2, ...x_N$ подается вектор входных значений $X = (x_1, x_2, ...x_N)$. На основании этих входных значений нейрон принимает решение, которое в данном случае, поскольку нейрон всего один, будет бинарным – «Да» (логическая единица) или «Нет» (логический 0). Решение принимается на основании весов нейрона w_i, которые являют собой некоторую дискретную модель памяти. Следовательно, каждое входное значение x_i умножается на

соответствующий вес w_i. Обычно веса при первом запуске необученной нейронной сети – это случайные дробные значения в диапазоне [-0.3…0.3]. Далее, полученные произведения $x_i w_j$ суммируются в i-ом нейроне

$$net_i = \sum_{j=1}^{M} x_i w_j \qquad (2.1)$$

После этого сумма подается на функцию активации, которой может служить пороговая функция (см. рис. 2.2)

$$f(net) = \begin{cases} 1, & net \geq \theta \\ 0, & net < \theta \end{cases} \qquad (2.2)$$

или вероятностная функция, такая как сигмоид (см. рис. 2.3)[102]

$$f(net) = \frac{1}{1+e^{-net}} \qquad (2.3)$$

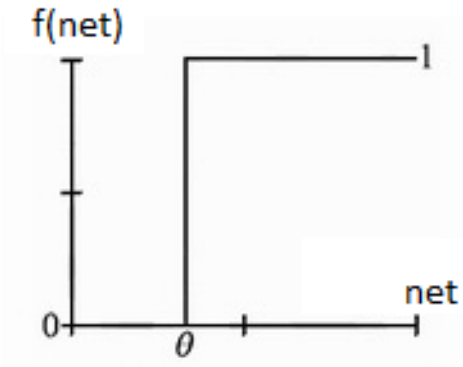


Рисунок 2.2. Пороговая функция

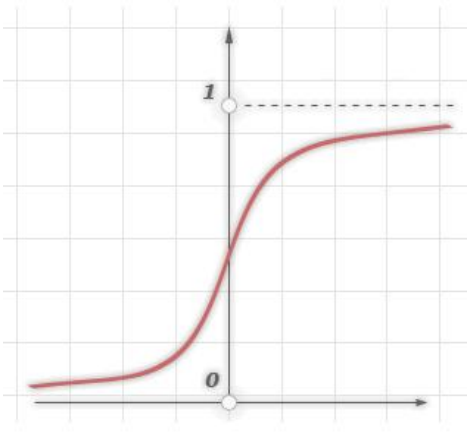


Рисунок 2.3. Сигмоидальная функция

В результате, на выходе будет получен ответ – 1 или 0.

Мультинейронная модель или модель с несколькими персептронами называетсяоднослойный персептрон и может решать задачи намного более сложные, чем приведенная ранее, простейшая модель. Однослойный персептрон представляет собой слой персептронов, где в каждый нейрон направлена посылка i-го входного значения x_i со встречным произведением на вес w_{ji}, соответствующий каждому j-му нейрону [122]. Данная модель представлена на рис. 2.4.

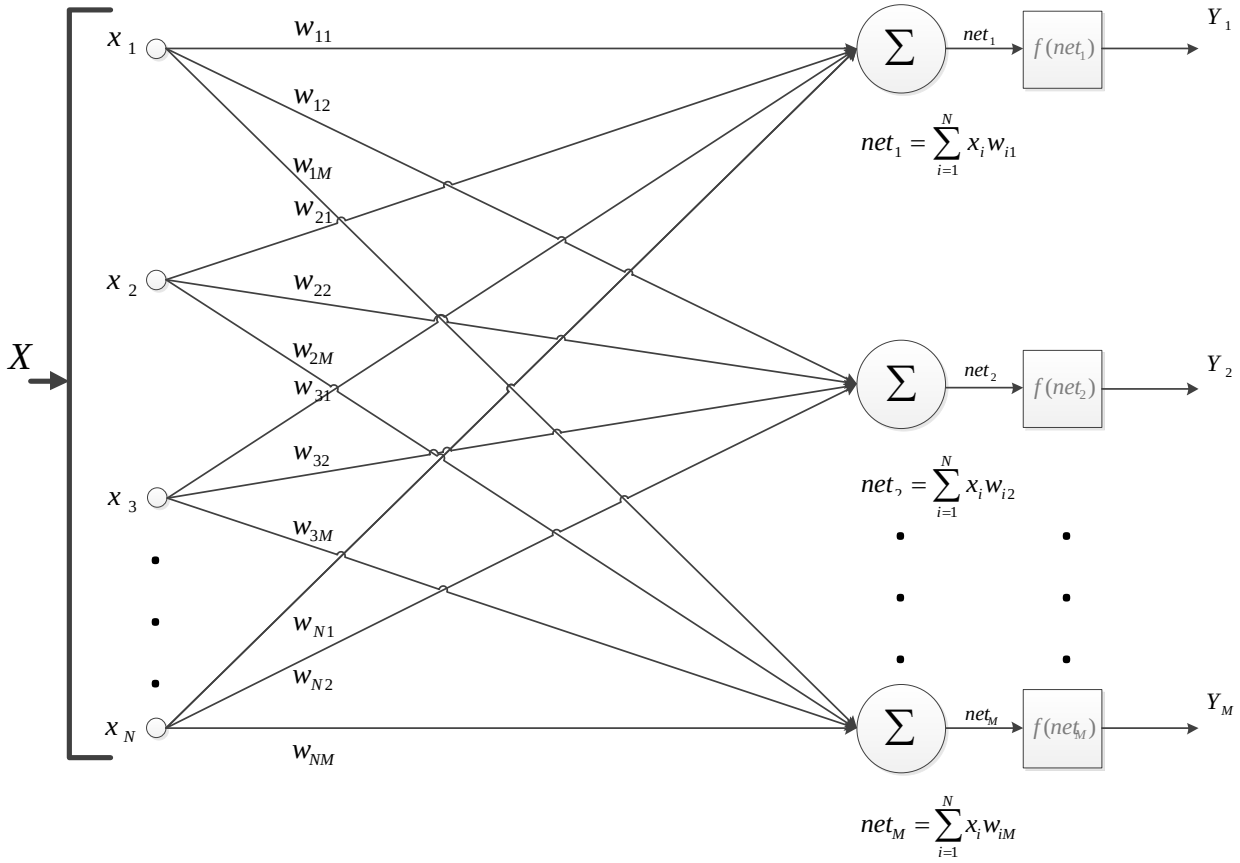


Рисунок 2.4 Однослойный персептрон. Общий вид.

Нейронная сеть может применяться для решения таких задач как [122]:

- классификация образов;
- кластеризация/категоризация объектов;
- аппроксимация функций;
- предсказание/прогноз;

- оптимизация;
- память, адресуемая по содержанию;
- управление.

Подобно биологической нейронной системе ИНС является вычислительной системой с огромным числом параллельно функционирующих простых процессоров с множеством связей. Модели ИНС в некоторой степени воспроизводят "организационные" принципы, свойственные мозгу человека [110].

Современные цифровые вычислительные машины превосходят человека по способности производить числовые и символьные вычисления. Однако человек может без усилий решать сложные задачи восприятия внешних данных (например, опознание человека в толпе только по его промелькнувшему лицу) со скоростью и точностью, превосходящей все существующие на сегодняшний день компьютерные системы. ИНС стремится повторить такой принцип вычислений человеческого мозга, моделируя его работу. Архитектура биологической нейронной системы совершенно не похожа на архитектуру машины фон Неймана (Таблица 2.1) [75, 78].

Таблица 2.1. Машина фон Неймана по сравнению с биологической нейронной системой.

	Машина фон Неймана	**Биологическая нейронная система**
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализована	Распределенная
	Адресация не по	Адресация по

	содержанию	содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	Хранимые программы	Самообучение
Надежность	Высокая уязвимость	Живучесть
Специализация	Численные и символьные операции	Проблемы восприятия
Среда функционирования	Строго определенная	Плохо определенная
	Строго ограниченная	Без ограничений

2.1.1. Симбиоз ИНС и БСС

Симбиоз двух технологий – БСС и ИНС может быть осуществлен в двух аспектах – прикладном и концептуальном [91, 123]:

1) Прикладной аспект рассматривает ИНС в качестве инструмента для решения узконаправленных, вспомогательных задач (кластеризация, прогнозирование, управление, и т. д.) для обеспечения функционирования БСС как сети передачи данных [91].

Задачи, которые могут решать ИНС в беспроводных сенсорных сетях:

- Кластеризация – одним из методов продления жизненного цикла сети, а также способов передачи пакетов, является метод кластеризации, который используется во многих протоколах маршрутизации БСС. Кластеризация может быть выполнена нелинейно – с помощью ИНС, которая оптимальным образом может разделить сенсорные узлы на кластеры в соответствие с заданными критериями (уровни радиовидимости, географические координаты, уровни остаточной энергии и др.).
- Маршрутизация – маршрутизация может выполняться посредством искусственного интеллекта. Один из примеров использования – многопутевая маршрутизация, где с помощью ИНС оценивается по

ряду критериев наиболее оптимальный с точки зрения скорости доставки данных, маршрут.

- Предсказание жизненного цикла сенсорной сети – жизненный циклсети можно оценить по ряду характеристик её узлов и по типу, а также частоте передачи данных. ИНС может выполнять аппроксимацию работы всей сети, чтобы предсказать, какие характеристики будут как у всей сети, так и у отдельных её компонентов в определенный период времени её функционирования.
- Анализ и управление трафиком – ИНС способна производить анализ траффика, производя его кластеризацию, тем самым, определяя, какой тип данных передается. На основании этого, нейронная сеть может принимать решения о сужении канала, либо обеспечении необходимой полосы пропускания.
- Обнаружение ошибок в сети - задается набор индикаторов ключевых параметров, по которым нейронная сеть сверяет с течение времени их значения с идеальными, на множестве которых было произведено обучение ИНС. На основании сравнения, нейронная сеть решает, работает ли сеть эффективно для выполнения поставленной задачи.

2) Концептуальный аспект - БСС может быть рассмотрена в качестве ИНС, если положить, что каждый из входов ИНС получает значения с помощью сенсорных узлов, которые преобразуют величины физического мира в электрические сигналы, которые оцифровываются и, далее, поступают на программно-моделируемые нейроны в качестве вектора входных величин (см. рисунок 2.5). Такая нейронная сеть будет принимать решения на основании входных значений БСС, которая служит мобильным, независимым, самоорганизующимся органом связи нейронной сети с реальным миром [91].

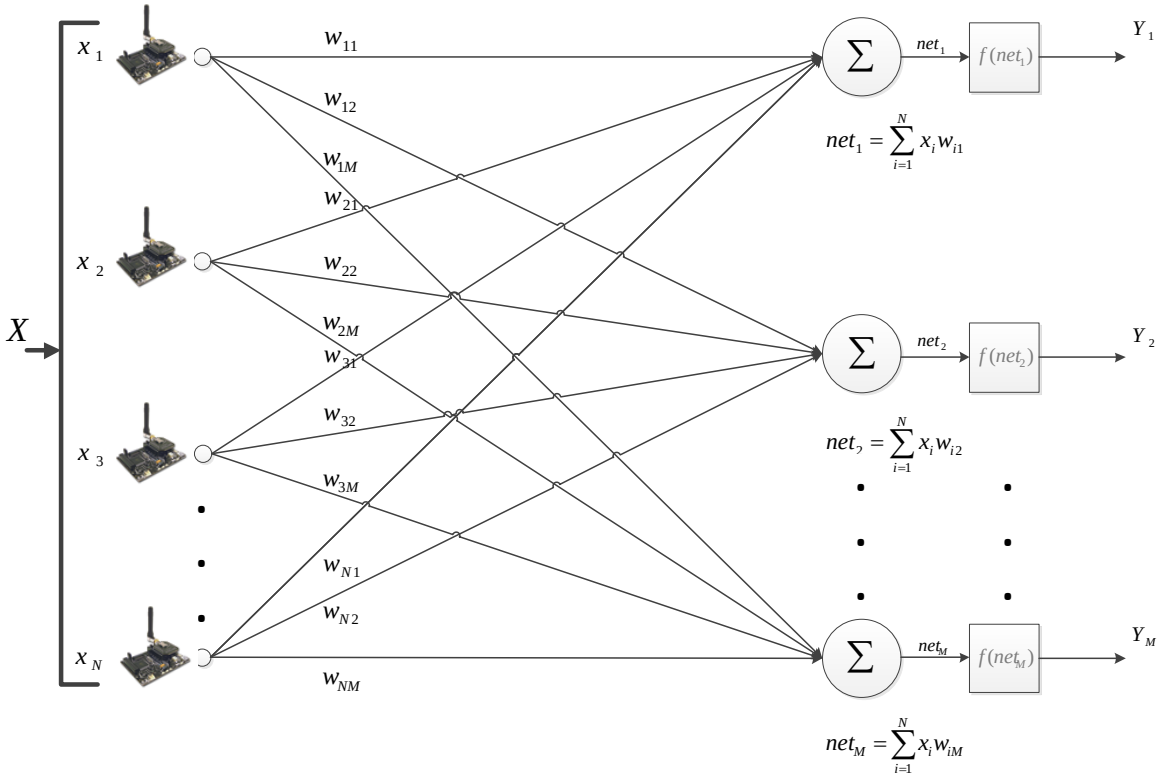


Рисунок 2.5 Нейронная беспроводная сенсорная сеть

Например, такой орган как глаза – функция, напрямую связана с задачей распознавания образов, может получать данные в виде закрашенности пикселей с определенным разрешением картинки посредством БСС. На основании этих данных, нейронная сеть может выделять образы из картинки, учитывая также искажения.

Таким образом, это будет нейронная беспроводная сенсорная сеть (НБСС), которая будет первым шагом на пути к созданию полноценного искусственного интеллекта, где его органы связи с внешним миром могут представляться в виде своих дискретных моделей ИНС, получающих данные посредством БСС [91].

Для данной диссертационной работы наиболее интересен прикладной аспект применения ИНС, поскольку данное направление исследований позволит решить ключевые проблемы, обозначенные выше.

2.1.2. Сходимость и производительность нейронных сетей

Теорема сходимости перцептрона — это теорема описанная и доказанная Ф. Розенблаттом.Она показывает, что элементарный перцептрон, обучаемый по методу коррекции ошибки (с квантованием или без него), независимо от начального состояния весовых коэффициентов и последовательности появления стимулов всегда приведет к достижению решения за конечный промежуток времени [122].

Таким образом, нейронная сеть найдет решение за конечный промежуток времени. Если же промежуток времени будет большим, то работу нейронной сети необходимо корректировать, пока не будут достигнуты удовлетворительные результаты.

Производительность нейронных сетей является одним из преимуществ, благодаря которому они имеют множество перспектив своего применения. Основная задача нейронных сетей — обработка образов. При этом у них, как и в человеческом мозге, отсутствуют общие шины, нет разделения на активный процессор и пассивную память, а вычисления и обучение распределены по всем элементарным процессорам — нейронам, которые функционируют параллельно. За счет этого нейронные сети позволяют добиться высокой производительности, которая может в миллионы раз превышать производительность традиционных компьютеров с последовательной архитектурой.

2.3. Исследование архитектур ИНС для кластеризации узлов БСС

В данной диссертационной работе рассматривается прикладной аспект применения ИНС в БСС, а именно, кластеризация беспроводных узлов. Данный аспект исходит из положения о том, что наиболее эффективными являются иерархические протоколы БСС, для работы которых необходимо наличие разделенных на кластеры узлов. Эффективность, соответственно, в конечном итоге зависит от оптимального деления узлов на кластеры.

Для эффективной кластеризации БСС произведено исследование относительно выбора архитектуры ИНС, которая бы позволяла правильно интерпретировать множество входных данных $X=(x_1,x_2,...x_N)$об узлах БСС и учитывала бы многопараметричность отличий каждого узла по отношению друг к другу. Следовательно, необходима архитектура ИНС, способная самостоятельно, без предварительного задания, выделять кластеры в данных, обеспечивающая высокое быстродействие при работе с большим количеством входных параметров (признаков) В этой связи были проанализированы 3 наиболее известных семейства архитектур ИНС, обучающиеся «без учителя», то есть автоматически выделяющие кластеры в данных.

1.3.1. Сети адаптивной резонансной теории

Сети АРТ, основоположником которых является С. Гроссберг, позволяют решить известную дилемму стабильности-пластичности. Суть дилеммы заключается в том, что нейронная сеть по мере своего развития должна быть способной к восприятию новых данных и созданию на основе этого новых классов - быть пластичной; в то же время сеть должна быть также стабильной, то есть при восприятии новых образов, существующие классы не должны искажаться и разрушаться [41].

Данные ИНС позволяет решить описанную дилемму, и являются векторным классификатором. Входной вектор $X=(x_1,x_2,...x_N)$ классифицируется *функцией сходства* $T(X,K_j)$в зависимости от того, на какой из классов K_j он наиболее похож:

$$T(X,K_j)=\frac{\|X\cap K_j\|}{\alpha+\|K_j\|},\alpha>0, \quad (2.4)$$

где α - параметр выбора.

$M(X, K_j)$ - называется *функцией совпадений*, которая определяет, насколько качественно входной вектор X относится к классу K_j:

$$M(X, K_j) = \frac{\|X \cap K_j\|}{\|X\|} \qquad (2.5)$$

Данная функция используется в сочетании с параметром $\rho \in (0,1]$, где $M(X, K_j) > \rho$ означает хорошее совпадение (резонанс).

Вектор избранного класса будет изменяться в соответствии с входным вектором так, чтобы более приблизиться по сходству к входному вектору с помощью *функции обновления* $U(X, K_j)$:

$$U(X, K_j) = (1-\beta)K_j + \beta(X \cap K_j) \qquad (2.6)$$

где β - скорость обучения, задаваемая в следующих пределах:

$$0 < \beta \le 1 \qquad (2.7)$$

Чем большие значения задаются в качестве β, тем выше скорость обучения. Самое быстрое обучение осуществляется при $\beta = 1$.

Если же входной вектор не похож ни на один из существующих классов, то создается новый класс, идентичный входному вектору.

Упрощенная структура сети АРТ представлена на рис. 2.6 в виде 5 функциональных модулей. Она включает два слоя нейронов, так называемых «слой сравнения» и «слой распознавания». Приемник 1, Приемник 2 и Сброс обеспечивают управляющие функции, необходимые для обучения и классификации [108].

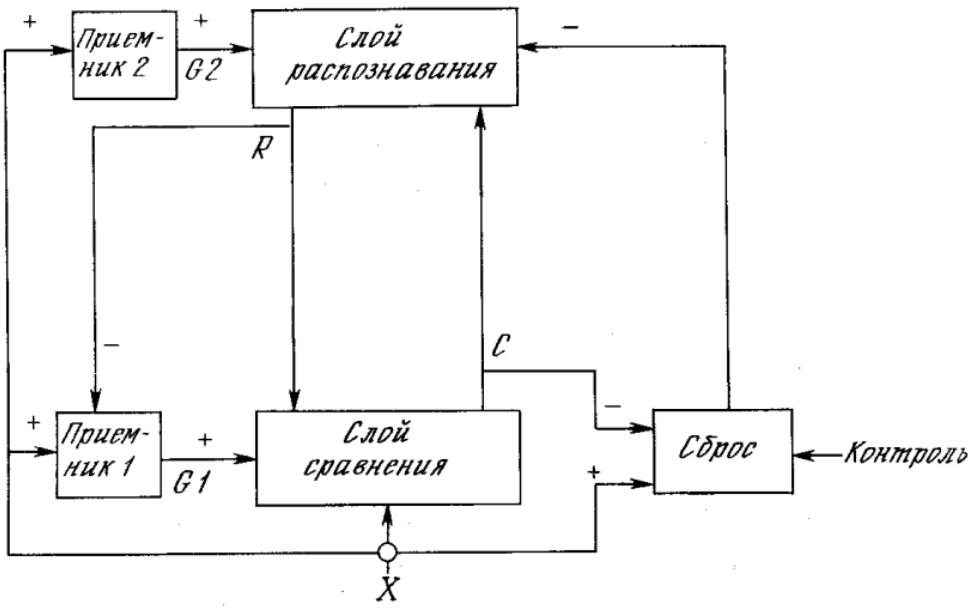


Рисунок 2.6. Упрощенная структура сети АРТ.

Среди сетей АРТнаиболее известны АРТ1, АРТ2, АРТ3 иFuzzyAPT. Сеть АРТ1 может работать только с логическими векторами, где входные образы описываются последовательностью единиц и нулей. Сети АРТ2 иАРТ3 позволяют работать с векторами из целых и дробных чисел. Сети FuzzyARTпредставляют собой расширенные сети АРТ1, путем ввода нечетких элементов. В результате FuzzyART могут работать с действительными значениями векторов.

Существуют также и некоторые модификации представленных выше сетей, например, сети АРТ2-апроизводят кластеризацию быстрее АРТ2, а результат у обоих сетей идентичен. Сети ARTMap могут обучаться по методу «с учителем» и «без учителя», для этого требуется комбинация из двух сетей семейства АРТ [66].

СемействосетейАРТимеет следующие недостатки:

- локализованность памяти – разрушение одного из нейронов ведет к потере информации обо всем классе;
- неконтролируемый рост нейронов – в ходе обучения сетей, может возникнуть неконтролируемая генерация классов, которая может быть вызвана шумами и искажениями входных данных [103].

Применение таких ИНС в БСС возможно, но проблема неконтролируемого роста нейронов является критической для применения в этой области. Неконтролируемое, бесконечное создание новых классов приведет к тому, что сходимость алгоритма обучения никогда не будет достигнута. Помимо этого, сети АРТ имеют сложный в реализации математический аппарат, и может потребоваться очень много времени для его отладки и реализации [109].

2.3.2. Неокогнитрон

Неокогнитрон был предложен К. Фукушимой и является многослойной иерархической сверточной сетью, усовершенствованной моделью сети Когнитрон [15]. Сеть состоит из иерархии нейронных слоев, каждый из которых является массивом плоскостей. Элемент массива состоит из пары плоскостей нейронов. Первая плоскость состоит из простых нейронных клеток, которые получают сигналы от предыдущего слоя и выделяют определенные образы. Эти образы далее обрабатываются сложными нейронами второй плоскости, задачей которых является сделать выделенные образы независимыми от их положения в пространстве. Структура сети Неокогнитрона и типового слоя представлена на рис. 2.7 и 2.8 [108, 109].

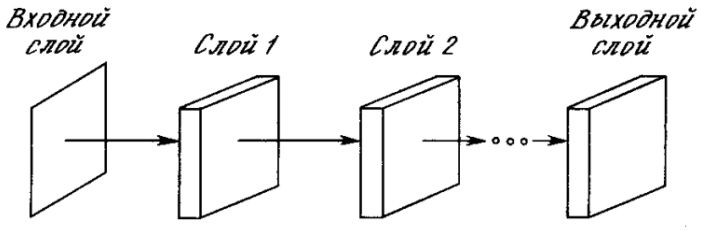


Рисунок 2.7. Структура сети Неокогнитрона.

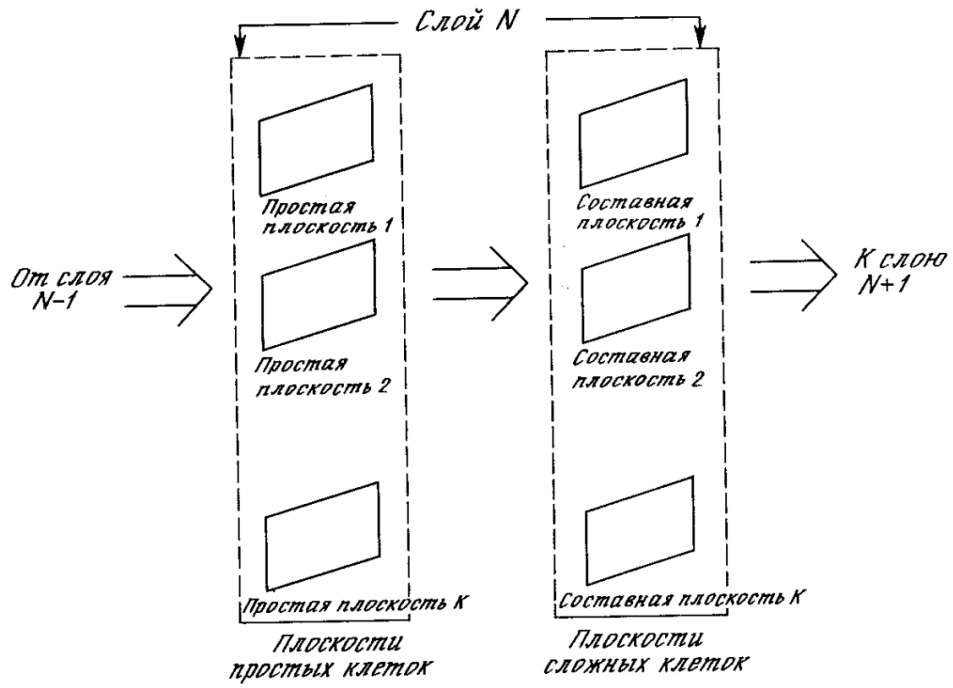


Рисунок 2.8. Структура слоя Неогонитрона.

Выходное значение нейрона S-слоя определяется следующим образом:

$$u_{Sl}\left(\vec{n},\vec{k}\right)=\frac{\theta_l}{1-\theta_l}\varphi\left(\frac{1+\sum_{k=1}^{K_{C_{l-1}}}\sum_{\left|\vec{v}\right|<A_{S_l}} a_{S_l}\left(\vec{v},\kappa,k\right)u_{C_{l-1}}\left(\vec{n}+\vec{v},\kappa\right)}{1+\theta_l b_{S_l}(k)u_l(n)}-1\right),\theta_l>0, \qquad (2.8)$$

$$u_l(n)=\sqrt{\sum_{k=1}^{K_{C_{l-1}}}\sum_{\left|\vec{v}\right|<A_{S_l}} c_{S_l}(v)\left[u_{C_{l-1}}\left(\vec{n}+\vec{v},k\right)\right]}, \qquad (2.9)$$

где θ_l - пороговая константа S-слоя; $b_{S_l}(k)$ - вес торможения, меняющийся во время обучения $a_{S_l}\left(\vec{v},\kappa,k\right)$ - вес связи предыдущего C-слоя, которые претерпевает изменения во время обучения; c_{S_l} - фиксированный вес; $\vec{n}$ - радиус вектора центра области; $\vec{v}$ - вектор окрестности центра области; $\varphi(x)=\max(0,x)$.

На основании вышеизложенного, а также [108], при высоких потребностях в исключении зависимости от пространственного положения входных образов Неокогнитрон может представлять очень массивную и ресурсоемкую структуру. Так происходит потому, что нейроны каждой пары плоскостей обучаются реагировать на определенный образ, представленный в определенной ориентации. Для другой ориентации или для нового угла поворота образа требуется новая пара плоскостей [103].

Применение Неокогнитрона эффективно при распознавании символов, имеющих шумы и сильные искажения. В задаче кластеризации БСС такие сети будут неэффективны в виду своей чрезмерной ресурсоемкости. Количество вычислений во много раз превышает приведенные выше сети и является достаточно большим. Это накладывает существенное ограничение на использование этой нейронной сети в виду высоких требований к объему памяти и времени, необходимому для вычислений [103].

2.3.3. Сеть Кохонена

Сеть Кохонена - это однослойная сеть, каждый нейрон которой соединен с компонентами входного вектора [79]. Была создана Т. Кохоненом. Структурно сеть состоит из трех слоев (рис. 2.9):

1) Входной слой нейронов, содержащий компоненты входного вектора $X = \{x\} = x_1,..., x_N$.
2) Слой Кохонена, который состоит из линейных взвешенных сумматоров $net_i = \sum_{j=1}^{M} x_i w_j$ - произведений значений на входах x_i на соответствующие веса w_i, реализует основной математический аппарат ИНС.
3) Выходной слой $Y = \{y\} = y_1,..., y_M$, нейроны которого передают выходные сигналы, образующие Мкластеров. При этом, чем больше сигнал нейрона, тем больше подобие входного вектора, весам нейрона.

На выходе сети обычно ставится интерпретатор, определяющий по величине сигнала, к какому классу относится входной вектор[79].

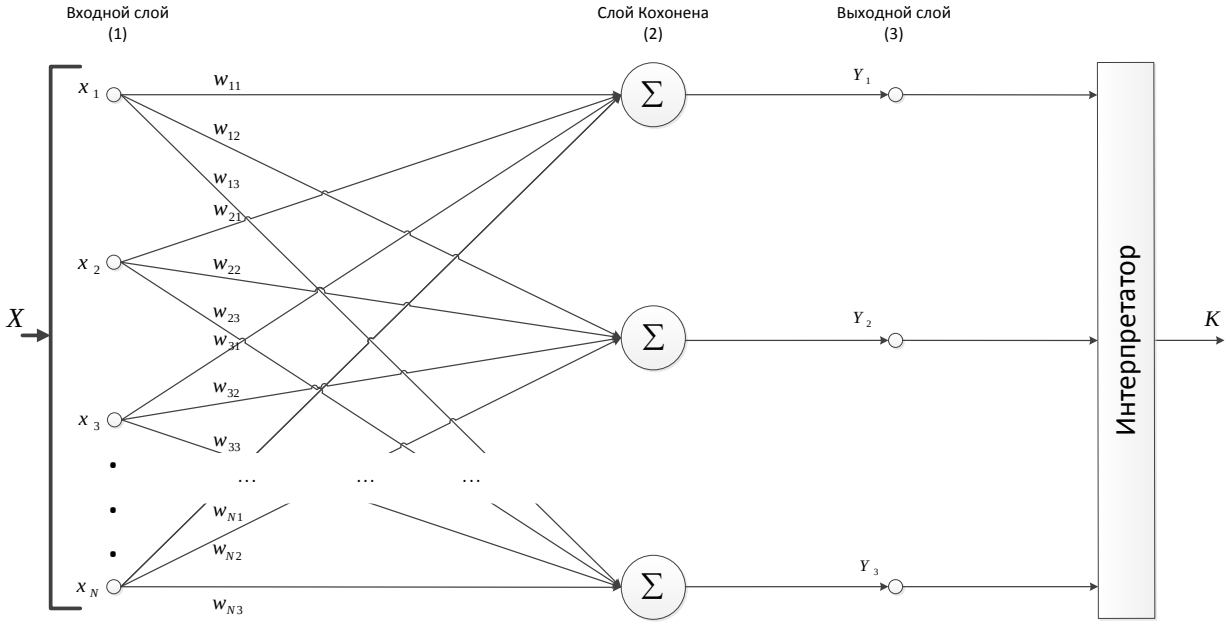


Рисунок 2.9. Структура сети Кохонена из Nвходных нейронов и 3 нейронов слоя Кохонена.

Преимуществами СКК являются [87]:

- быстрое обучение;
- возможность визуализациимногомерных входных данных в виде двумерной карты Кохонена [76];
- возможность упрощения многомерной структуры.

Для обучения Сети Кохонена используют одно из правил (методов) обучения. Классическая ИНС обучается по правилу: «Победитель забирает все» (WinnerTakesAll, WTA). Это правило еще называют правилом жесткой конкуренции:во время обучения производится соотношение каждого входного вектора $x_i \in$ с ближайшим кластером:

$$d\left(x, w_j\right) = \arg\min_{j \in Y} d\left(x, w_j\right) \qquad (2.10)$$

Существуют также и некоторые другие известные правила обучения:

- Правило справедливой конкуренции CWTA [68]:

$$d(x, w_j) = \arg\min_{j \in Y} C_j d(x, w_j), \qquad (2.11)$$

где C_j - количество побед j-го нейрона в ходе обучения.

Таким образом, нейроны «отдыхают» в случае слишком частого отношения входных объектов к своему кластеру.

- Правило мягкой конкуренции WTM[68].

Математический аппарат, скорость и качество обучения ИНС во многом определяет выбранное правило обучения. Приведенные выше правила обучения эффективны при решении большинства задач с помощью Сети Кохонена, но они имеют ряд недостатков, которые могут проявляться в зависимости от специфики решаемой задачи:

- Зависимость от инициализации весов. Значение весов, которые нейроны приобрели при старте сети, имеют значительное влияние на ход обучения: количество кластеров, расположение.
- Фиксированное число кластеров. Сети необходимо заранее задать число кластеров, на которые будут разбиты данные.
- Искажения весов нейронов. Данная особенность не всегда проявляется в сетях Кохонена, а только при случайной инициализации весов. В результате, близкие друг к другу по параметрам узлы могут оказаться далекими. Так кластеры могут неоправданно разрываться другими. И наоборот, далекие друг для друга узлы могут оказаться рядом. Это объясняется тем, что начальная инициализация весов случайными значениями исказила наборы весов кластеров сети по отношению к подаваемой на вход сети при обучении выборке. Помимо этого влияние на усиление искажений оказывает и ограниченное количество кластеров, к которым сеть Кохонена может отнести тот или иной объект, не имея возможности создавать новые кластеры. Таким образом, возникают искажения, которые являются неизбежными для классической модели сети Кохонена при решении ряда задач [68].

Устранение всех этих недостатков возможно при использовании нестандартного правила обучения нейронной сети – Конструктивного метода [64]. Данное правило позволяет избежать некорректной инициализации весов, так как изначально сеть будет состоять лишь из одного нейрона $K_1 = \{w_1\} = w_{11},..., w_{1N}$, веса которого будут назначены согласно значениям первого входного обучающего вектора $X_1 = \{x_1\} = x_{11},..., x_{1N}$:

$$K_1 = X_1, \qquad (2.12)$$

$$\{w_1\} = \{x_1\}, \qquad (2.13)$$

$$w_{11} = x_{11},..., w_{1N} = x_{1N} \qquad (2.14)$$

Далее, сеть Кохонена будет автоматически создавать новые классы, если входные вектора будут достаточно отличаться от уже имеющихся в соответствии с условием:

$$d(x, w_i) \le \sqrt{N}R, \quad (2.15)$$

где N - количество входов сети, а R – радиус чувствительности нейрона, $d(x, w_i)$ - расстояние между входным вектором $\{x\}$ и вектором весов $\{w_i\}$ в используемой метрике, как правило – это Евклидово расстояние:

$$d(x, w_i) = \|x - w_i\| = \sqrt{\sum_{j=1}^{N} (x_j - w_{ij})^2}, \qquad (2.16)$$

тогда среднеквадратическая ошибка равняется:

$$e_j = x_j - w_{ij} \qquad (2.17)$$

Элемент $\sqrt{N}$ в формуле является Евклидовым расстоянием, для которого среднеквадратическая ошибка является максимальной:

$$\sqrt{N} = \max(e_j) \quad (2.18)$$

Исходя из того, что входной вектор $\{x\}$, как правило, нормируется в пределах $[0,1]$, получим:

$$d(x, w_i) = \max(e_j \in [0,1]) = \sqrt{\sum_{j=1}^{N} 1^2} = \sqrt{N} \quad (2.19)$$

Использование данного правила обучения решает проблему фиксированного числа кластеров. При этом неконтролируемый рост нейронов исключается, а максимальное количество нейронов равняется количеству компонент входного вектора.

Конструктивный метод вносит в сеть такое свойство как радиус чувствительности нейронов, который позволяет задавать размер кластеров.

2.3.4. Выбор ИНС для кластеризации БСС

Для решения задач диссертационной работы выбрана ИНС Кохонена. Выбор обусловлен отсутствием недостатков, которые присущи сходным по методологии обучения без учителя нейронным сетям (неокогнитрону и семейству сетей АРТ). Имеющиеся недостатки ИНС Кохонена устраняются благодаря использованию в качестве метода обучения – Конструктивного метода.

Благодаря использованию Конструктивного метода [64] обучения ИНС Кохонена удается также превзойти по эффективности Сети адаптивной теории резонанса АРТ, поскольку последние имеют весомый недостаток – неконтролируемый рост нейронов в ходе обучения.

Для возможности учета особенностей входных векторов, описывающих беспроводные узлы, в Конструктивный метод предложено добавить условие связности:

$$\exists x_j \neq 0 \cap w_{ij} \neq 0 : i = j \,\&\, w_i \in d(x, w_i) \qquad (2.20)$$

Данное условие позволит определить имеет ли вектор, описывающий радиовидимость текущего беспроводного узла связь с нейроном-победителем, это позволит избежать разрывов кластеров и однозначно определить изолированные узлы и учитывать радиовидимость при кластеризации.

Выводы по главе 2

Производительность ИНС является одним из преимуществ, за счет которого ИНС имеют множество перспектив применения. Высокая производительность обуславливается их сходством со своим биологическим аналогом – мозгом человека. В ИНС также как и в нем, нет общих шин и отсутствует разделение на активный процессор и память. Все вычисления и обучение распараллелены по процессорам, созданным из примитивов – нейронам, функционирующим параллельно. Благодаря этому, ИНС позволяют достичь высокой производительности, превышающей в миллионы раз производительность традиционных компьютеров с последовательной архитектурой, а также существующие многоядерные системы [89].

Нелинейный анализ, выражающийся в решении задач многопараметрической оптимизации, позволяет ИНС осуществлять решение задач, в которых неизвестны закономерности между входными и выходными данными. Классические способы решения и экспертные системы в таких случаях не способны предоставить решение.

Кроме того, ИНС способна работать при наличии большого числа неинформативных шумовых данных. Нет необходимости делать их предварительный отсев, нейронная сеть может автоматически определить их малую значимость для решения задачи, и не будет учитывать их.

Одно из основных применений ИНС – это кластеризация данных. Как было выявлено на основании произведенного анализа в данной главе, использование ИНС в БСС возможно в двух аспектах – концептуальном и прикладном. Для исследования данной диссертационной работы наибольший интерес представляет прикладной аспект. В рамках данного аспекта, установлено, что кластеризацию узлов БСС возможно производить посредством ИНС

Существует множество архитектур ИНС, среди которых выбрана наиболее эффективная для использования в БСС. При выборе ИНС важным

было учесть наличие свойства самообучения, т.е. обучения «без учителя», поскольку при кластеризации БСС не существует никаких эталонных образов, а в качестве входных данных должна использоваться информация обо всех узлах БСС. Следовательно, ИНС должна сама выделить кластеры во входных данных.

Среди нейронных сетей, способных самостоятельно выделять кластеры в данных – обучаться «без учителя», наиболее известны и исследованы Самоорганизующиеся карты Кохонена (Kohonen’sself-organizingmap, SOM, СКК), Сеть адаптивной резонансной теории (Adaptiveresonancetheory, ART, АРТ)и Неокогнитрон.

Согласно исследованиям, описанным в [108] семействосетейАРТ – АРТ1, АРТ2, АРТ3 иFuzzyART – имеют недостаток, связанный с неконтролируемой генерацией новых нейронов и, следовательно, очень чувствительны к искажениям. Структура сети Неокогнитронакак утверждается висследованиях [103] является сложной, а количество вычислений во много раз превышает приведенные выше сети и является достаточно большим. Это накладывает существенное ограничение на использование этой нейронной сети в виду очень высоких требований к объему памяти и времени, необходимому для вычислений. Таким образом, выявлено, что наиболее оптимальным вариантом для кластеризации БСС является использование ИНС Кохонена.

Выбор ИНС Кохонена обусловлена отсутствием недостатков, которые присущи сходным по методологии обучения без учителя, нейронным сетям (неокогнитрону и семейству сетей АРТ). Имеющиеся недостатки ИНС Кохонена устраняются благодаря использованию в качестве метода обучения – Конструктивного метода.

Благодаря использованию Конструктивного метода [64] обучения ИНС Кохонена представляется возможным превзойти по эффективности Сети

адаптивной теории резонанса АРТ, поскольку последние имеют весомый недостаток – неконтролируемый рост нейронов в ходе обучения.

Для наибольшей эффективности Конструктивного метода и его адекватности относительно задачи кластеризации узлов БСС, введено дополнительное условие связности беспроводного узла с нейроном-победителем.

ГЛАВА 3. Способы кластеризации узлов и нейросетевой протокол маршрутизации БСС

В данной главе представлены практические результаты диссертационной работы. Предложена матрица радиовидимости, являющаяся математическим описанием связности узлов сети и радиовидимости каждого узла по отношению ко всем остальным узлам сети. Данное математическое представление позволит описать все узлы сети и подготовить к соответствующей форме данные для подачи на вход ИНС.

На основе ИНС Кохонена, обучаемой по Конструктивному методу [64].разработан способ нейросетевой кластеризации БСС [88]. Также разработан матричный способ кластеризации БСС.

Для эффективного использования разработанного способа нейросетевой кластеризации, разработан иерархический протокол маршрутизации данных в БСС.

3.1. Математическое описание узлов БСС для ИНС

Для работы нейросетевого и матричного способов кластеризации, необходимо предварительно представить данные в виде матрицы радиовидимости, которая является математическим описанием узлов БСС и отражает уровень радиовидимости между всеми узлами в процентном соотношении.

Каждый узел БСС q_i имеет свое множество соседей $Z_i = \{z\} = z_1,...,z_j,...,z_{L_i}, Z_i \in q_i$ где L_i - мощность множества соседей q_i-го узла. Множество может быть пустым в случае, если узел q_i является изолированным $Z_i \in \varnothing$. Каждому соседнему узлу z_j из множества Z соответствует уровень мощности получаемого сигнала p_j (RSS, ReceivedSignalStrength) [85] по отношению к узлу – владельцу множества

соседей - q_i . Таким образом, каждый узел q_i имеет свой вектор мощностей $\vec{p}$, где каждый уровень мощности p_j соответствует определенному соседнему узлу из множества Z_i :

$$\vec{p}_i \in Z_i = \{p\} \in \{z\} = p_1 \in z_1, ..., p_j \in z_j, ..., p_L \in z_L \qquad (3.1)$$

Следовательно,

$$\forall q_i \exists Z_i \in q_i : \left(p_1 \in z_1, ..., p_j \in z_j, ..., p_L \in z_L\right) \in q_i \qquad (3.2)$$

Матрица радиовидимостиP (матрица энергетической видимости, матрица радиосвязи, матрица мощностей) – это квадратная матрица порядкаN описывающая связность между узлами беспроводной сети, где каждому узлу q_i ставится в соответствие мощностей сигналов $\{p\}$(RSS) его соседей $\{z\}$(3.3):

$$P = \begin{pmatrix} \infty & p_{12} & ... & p_{1j} & ... & p_{1n} \\ p_{21} & \infty & ... & p_{2j} & ... & p_{n2} \\ ... & ... & ... & ... & ... & ... \\ p_{i1} & p_{i2} & ... & \infty & ... & p_{in} \\ .... & ... & ... & ... & ... & ... \\ p_{n1} & p_{n2} & ... & p_{nj} & ... & \infty \end{pmatrix}, \qquad (3.3)$$

$$\mathrm{p}_{ij} = \begin{cases} \mathrm{p}_{\mathrm{ij}}, \text{если } \mathrm{i} \neq \mathrm{j} \\ \text{иначе } \infty \end{cases}, \; p_{ij} \in [0\%...100\%], \qquad (3.4)$$

где p_{ij} – уровень мощности сигнала узла i по отношению к узлу j, ∞ – уровень мощности сигнала узла по отношению к самому себе.

Для сети из 7 узлов матрица радиовидимости Рбудет выглядеть следующим образом (3.5):

$$P = \begin{pmatrix} \infty & 60 & 80 & 0 & 0 & 0 & 0 \\ 60 & \infty & 65 & 0 & 0 & 0 & 0 \\ 80 & 65 & \infty & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \infty & 85 & 0 & 0 \\ 0 & 0 & 0 & 85 & \infty & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \infty \end{pmatrix} \quad (3.5)$$

Бесконечность на практике интерпретируется как 100% (3.6) для удобства машинной обработки:

$$P = \begin{pmatrix} \infty & 60 & 80 & 0 & 0 & 0 & 0 \\ 60 & \infty & 65 & 0 & 0 & 0 & 0 \\ 80 & 65 & \infty & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \infty & 85 & 0 & 0 \\ 0 & 0 & 0 & 85 & \infty & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \infty \end{pmatrix} \underset{\infty\, def\, 100}{=} \begin{pmatrix} 100 & 60 & 80 & 0 & 0 & 0 & 0 \\ 60 & 100 & 65 & 0 & 0 & 0 & 0 \\ 80 & 65 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 85 & 0 & 0 \\ 0 & 0 & 0 & 85 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{pmatrix} \quad (3.6)$$

Каждая строка матрицы Рявляется входным вектором для нейронной сети и несет информацию о каждом из узлов сети – о его видимости относительно других узлов сети. С помощью данной матрицы можно описать граф $G = (Q, P)$, где Q – множество сенсорных узлов, P – множество уровней мощности сигнала (рис. 3.1):

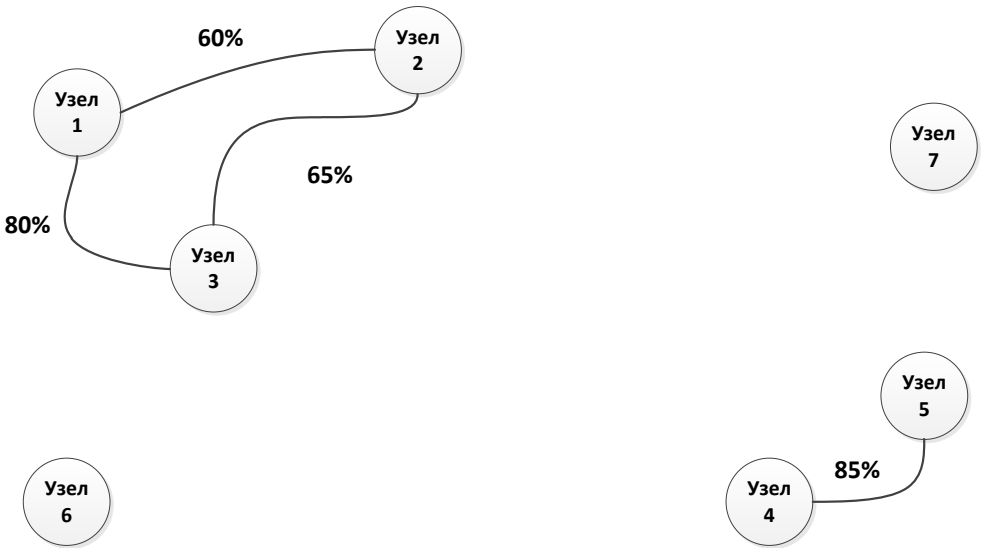


Рисунок 3.1. Граф радиовидимости для сети из 7 узлов

3.2. Способ нейросетевой кластеризации БСС.

Используем математическое описание узлов БСС $q_1,...q_N$ в виде матрицы радиовидимости P.

Для матрицы радиовидимости P осуществим нормализацию всех значений по следующей формуле (3.7):

$$p_{NORM} = \frac{p}{p_{MAX}}, \quad (3.7)$$

где $p_{MAX} = 100$.

Получим нормализованную матрицу радиовидимости (3.8):

$$P_{NORM} = \begin{pmatrix} 1 & \frac{p_{12}}{p_{MAX}} & \dots & \frac{p_{1j}}{p_{MAX}} & \dots & \frac{p_{1n}}{p_{MAX}} \\ \frac{p_{21}}{p_{MAX}} & 1 & \dots & \frac{p_{2j}}{p_{MAX}} & \dots & \frac{p_{n2}}{p_{MAX}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{p_{i1}}{p_{MAX}} & \frac{p_{i2}}{p_{MAX}} & \dots & 1 & \dots & \frac{p_{in}}{p_{MAX}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{p_{n1}}{p_{MAX}} & \frac{p_{n2}}{p_{MAX}} & \dots & \frac{p_{nj}}{p_{MAX}} & \dots & 1 \end{pmatrix} \quad (3.8)$$

Пусть параметры сети искусственной нейронной сети Кохонена NET будут заданы функционалом:

$$NET(N, K, \alpha_0, \tau, R, \varepsilon, \{L\}) \quad (3.9),$$

где:

- N– количествовходных нейронов;
- K–количество нейронов слоя Кохонена (выходных нейронов). Изначально значение равняется согласно конструктивному методу обучения, только лишь одному нейрону;
- α_0 – начальное значение скорости обучения;
- τ – постоянная времени обучения;
- R – радиус чувствительности;
- ε – точность обучения;

- $\{L\}$ – обучающая выборка.

Сети Кохонена (3.9) необходимо задать следующие входныепараметры:

$$NET\left(|\{q\}|,1,\alpha_0 = 0.7,\tau = 1000,R = [0.22;0.36],\varepsilon = 0.1,P_{NORM}\right) \quad (3.10),$$

где P_{NORM} - нормализованная матрица радиовидимости.

Количество входных нейронов зададим Nравным количеству узлов БСС $N = |\{q\}|$. Начальное значение скорости обучения α_0 и постоянная времени обучения заданы согласно рекомендациям[65] и скорректированы при моделировании ИНС. Значение радиусачувствительности R влияет на размер кластера, задание данной величины для корректной кластеризации БСС возможно только в пределах от 0.22 до 0.36 согласно критериям, приведенных рекомендациях [64] для классификации графов.Задание точности обучения ε = 0.1 является достаточным для определения стабильного состояния весов нейронов согласно [73]. В качестве входной выборки $\{L\}$ зададим все множество векторов нормализованной матрицы радиовидимости P_{NORM}, описывающей связность соседних узлов q_i.

Используем алгоритм обучения сети Кохонена по Конструктивному методу с настроенными коэффициентами функций для кластеризации узлов БСС.

Последовательно подаем на вход сети обучающие вектора $E_1 \ldots E_N$, где в качестве векторов будут выступать строки нормализованной матрицы радиовидимости P_{NORM} от 1 до Nсоответственно:

1) Подаем на вход сети обучающий вектор E_i.
2) Если это первый запуск сети после инициализации, то присваиваем единственному, существующему в сети нейрону значения поданного обучающего вектора, затем переходим, затем переходим к шагу 1. Иначе, переходим к шагу 2.

3) Определяем нейрон-победитель (нейрон с наименьшим расстоянием) для текущего вектора обучения. В качестве метрики - Евклидово расстояние, которое вычисляем по формуле (3.11):

$$d(x, w_i) = \|x - w_i\| = \sqrt{\sum_{j=1}^{n} (x_j - w_{ij})^2} \qquad (3.11)$$

4) Если Евклидово расстояние между нейроном-победителем и входным вектором $d(x, w_i)$ не удовлетворяет условию (3.12) или не выполняется условие связности беспроводного узла с нейроном-победителем (3.13), то в сеть добавляется новый нейрон, который принимает значение этого входного вектора и переходим к шагу 1.

$$d(x, w_i) \le \sqrt{N} R \qquad (3.12)$$

где N - количество входов сети, а R – радиус чувствительности нейрона.

$$\exists x_j \ne 0 \cap w_{ij} \ne 0 : i = j \,\&\, w_i \in d(x, w_i) \quad (3.13)$$

5) Подстраиваем веса (компоненты вектора) нейрона-победителя и близлежащих нейронов относительно текущего обучающего вектора E_i по формуле (3.14):

$$w_i^{t+1} = w_i^t + \cdot\alpha(t) \cdot \left(x - w_i^t\right) \quad (3.14)$$

где w_i^t - вектор весов i-го нейрона, t – номер итерации обучения, x - входной вектор, $\alpha(t)$ - коэффициент скорости обучения.

На данном этапе происходит подстройка весов всех нейронов Кохонена. Величина изменения весов w_i^k каждого нейрона зависит от произведения разности каждого веса w_i^k и компонента входного вектора x на коэффициент скорости обучения $\alpha(t)$:

$$\alpha(t) = \alpha_0 \cdot \exp\left(-\frac{t}{\tau}\right) = 0.7 \cdot \exp\left(-\frac{t}{1000}\right) \qquad (3.15)$$

6) Если текущий вектор - последний вектор обучения E_N, то проверяем, изменились ли веса каждого нейрона больше, чем на ε по отношению к

предыдущему состоянию, после предыдущего прохождения последнего обучающего вектора. Если нет, то завершаем обучение сети. Если да, то переходим на первый шаг и продолжаем обучение с первого обучающего вектора E_1.

После обучения, назначаем в качестве ГКУ центроиды каждого кластера.

3.2. Матричный способ кластеризации БСС.

Данный способ основан на измерении мощностей сигналов между каждым узлом БСС. Используем математическое описание узлов в виде матрицы радиовидимости Р.

Способ заключается в поиске по матрице радиовидимостиРэлемента с наибольшим уровнем мощности $\max(p_{i,j})$ путем перебора строк и столбцов. После нахождения такого элемента вокруг него будет образовываться кластер, путем поиска соседей. Если найдено несколько одинаковых уровней мощности – максимальных из всей матрицы, то искомым элементом будет тот, который был найден раньше всех. Поиск соседей заключается в измерении разниц в мощностях от выбранного элемента по отношению ко всем остальным элементам (узлам). Далее, находится среднее арифметическое значение разниц мощностей. После чего, все узлы, мощности которых меньше этого среднего арифметического значения, попадают в один кластер. Узлы, вошедшие в образовавшийся кластер, будут далее исключены из рассмотрения. Алгоритм будет повторяться до тех пор, пока не закончатся узлы, для которых не образован кластер. Максимальное количество узлов, которое может войти в кластер перед началом кластеризации ограничивается константой N_K.

Пусть имеется некоторая матрица радиовидимостиР (3.3).

1) Найдем первую пару узлов (элемент), между которыми мощность максимальна относительно других пар в соответствие с матрицей Р.

Обозначим такой элемент как $\mathrm{Pmax}_1 = \max(\mathrm{p}_{\mathrm{i,j}})$. Далее, для данного элемента будет построен вектор$V1_{max\,(p_{ij})}$ $V1_{\max(\mathrm{p}_{\mathrm{i,j}})}$ из уровней мощностей до всех остальных $k = (n - P\max_1)$ из n узлов сети и отсортирован по убыванию мощности.

$$V1_{\max(\mathrm{p}_{\mathrm{i,j}})} = \left(\mathrm{Pmax}_1; p_{ij}^1 < P\max_1; p_{ij}^2 < p_{ij}^1; p_{ij}^3 < p_{ij}^2, \ldots p_{ij}^k < p_{ij}^{k-1}\right) \qquad (3.16)$$

2) Вычислим разницы мощностей между всеми узлами:

$$\mathrm{Pmax}_1 - p_{ij}^1; p_{ij}^1 - p_{ij}^2; p_{ij}^2 - p_{ij}^3, \ldots p_{ij}^k - p_{ij}^{k-2} \qquad (3.17)$$

3) Найдем среднее арифметическое этих разниц D_1 для вектора $V1_{\max(\mathrm{p}_{\mathrm{i,j}})}$:

$$D_1 = \frac{\sum_{l=1}^{k} p_{ij}^l}{k} \qquad (3.18)$$

4) Добавим в первый кластер K_1 узлы с мощностью Pmax_1 и все узлы вектора $V1_{\max(\mathrm{p}_{\mathrm{i,j}})}$, у которых мощность $\mathrm{p}_{\mathrm{i,j}}$ меньше D_1. Добавление узлов производится до тех пор, пока количество узлов кластера не превысит $\mathrm{N_K}$. Если $\mathrm{N_K}$= 0, то максимальное количество узлов для кластера не ограничивается.
5) Исключим из рассмотрения столбцы и строки тех узлов матрицы, которые вошли в сформированный кластер. Повторим алгоритм с шага 1. Повторение будет производиться до тех пор, пока не останется узлов, которым не назначены кластеры.

3.2.1. Матричный (жадный) способ кластеризации БСС.

Разработана также модификация матричного метода кластеризации, отличающаяся от базового способа тем, что в кластер входят абсолютно все узлы, у которых мощность по отношению паре узлов с максимальной мощностью не равна нулю. Количество узлов, которые могут войти в кластер также ограничивается задаваемой перед началом кластеризации константой $\mathrm{N_K}$.

3.3. Протокол нейросетевой маршрутизации БСС.

На основе способа нейросетевой кластеризации разработан протокол нейросетевой маршрутизации БСС - EDNCP (Energydistanceneuralclusteringprotocol, протокол энергетических расстояний нейросетевой кластеризации). Данный протокол предназначен для работы в БСС. Основная цель EDNCP–маршрутизация данных в сети, разделенной на кластеры с наличием резервных путей передачи данных [123].

В качестве физического уровня и уровня управления доступом к каналу (MAC) выступает стандарт IEEE 802.15.4: диапазон 2.4 ГГц, 16 каналов шириной 5 МГц, CSMA/CA (в дальнейшем возможно рассмотрение применения протокола MAC основанного на TDMA).

Предварительные условия

Узлы дислоцируются на целевую область и пребывают в режиме периодического засыпания (время сна и бодрствования возможно установить экспериментально или сделать его случайным). Во время «бодрствования» радиоэфир прослушивается на предмет появления в нем БС. Сенсоры отключены, так как считаем, что сбор данных не нужен при отсутствии БС – средства сбора и управления.

Описание технологии.

Использование протокола EDNCP предусматривает наличие БС, которая становится центральным узлом, на который передаются данные со всех узлов сети. В протоколе имеются 3 основные структуры данных.

Первая структура называется PowerMatrix, хранит Матрицу радиовидимостиР, которая создается в процессе инициализации сети и служит для разделения сети на кластеры посредством Способа нейросетевой кластеризации. После кластеризации узлы каждого кластера передают свои данные на БС посредством ГКУ.

Вторая и третья структура хранится на всех узлах сети. Третья структура только на подконтрольных ГКУ узлах.

Вторая структура называется ToBSPath (табл. 3.1), хранится на всех узлах сети и содержит записи об узлах, через которые можно передать пакеты данных на БС. Записи состоят из 3 полей – ID, LT и CH, где

- ID - идентификатор узла;
- LT (Life Time) – время жизни пакета, фактически – количество совершенных хопов от БС до определенного узла;
- CH(ClusterHead) – статус узла в качестве ГКУ (0 – не ГКУ, 1 – ГКУ).

Структура используется для выбора узла, через который на БС будут передаваться данные в соответствие с приоритетом: наличие статуса CH и наименьшее значение LT; в случае отсутствия узла со статусом CH, будет динамически назначаться ретранслятор – узел с наименьшим LT.

Таблица 3.1. Пример структуры ToBSPath

ID	**LT**	**CH**
2	200	0
4	100	1
5	50	0

Третья структура называется ToCHPath (табл. 3.2), хранится только на подконтрольных определенному ГКУ узлах сети и содержит записи об узлах, через которые можно передать пакеты данных на ГКУ. Записи состоят из 2 полей – ID, LT, назначения полей аналогично структуре ToBSPath. Структура используется для выбора узла, через который на ГКУ будут передаваться данные – выбирается узел с наименьшим значением LT.

Таблица 3.2. Пример структуры ToCHPath

ID	**LT**
2	200
4	100
5	50

Все узлы сети производят передачу данных. Пакеты данных имеют своего адресата – ГКУ или БС.

Подконтрольные узлы могут назначать в качестве адресата только ГКУ, но передавать данные они могут как предназначенные для ГКУ, так и для БС в зависимости от назначенного адресата. В первом случае для передачи используется структура ToCHPath, во втором – ToBSPath.

ГКУ могут назначать пакету данных в качестве адресата только БС, при этом передавать пакеты они могут только предназначенные для БС. Если им передают данные, предназначенные для ГКУ, где в качестве ГКУ выступают они сами, то они меняют адресата на БС.

Правила функционирования до кластеризации

1) БС посылает hello-пакеты всем узлам сети, находящимся в пределах ее радиовидимости. В hello-пакет записывается параметр LT = 0.
2) Каждый узел, получивший hello-пакет, записывает во внутреннюю структуру ToBSPathпараметр LT пакета и ID узла, от которого он пришел. Структура ToBSPathна данный момент содержит записи только с двумя полями - ID и LT узла (табл. 3.1). Третье поле – CHбудет задействовано на этапе назначения ролей ГКУ.
3) Получив hello-пакет, узел продвигает его дальше по сети только в том случае, если LTэтого пакета меньше всех остальных, пришедших на данный узел согласно структуре ToBSPath. В противном случае, только фиксируются параметры пакета LTиIDузла, от которого пришел пакет без дальнейшего реплицирования по сети. Это необходимо для того, чтобы выявить оптимальные маршруты передачи до БС, а для инициализации сети достаточно продвижения через узел хотя бы одного пакета.
4) Получив hello-пакет, узел отправляет на БС power-пакет, который содержит уровень мощности сигнала от узла, передавшего hello-пакет.
5) Узлы, получающие power-пакет, передают его на БС согласно своей внутренней структуре ToBSPath– для передачи выбирают соседний

узел с наименьшим значением LT, который далее действует аналогично.

6) Если power-пакеты перестают приходить на БС в течение заданного времени PowerTimeout, то запускается процесс кластеризации узлов на БС. После процесса кластеризации начинают действовать Правила функционирования при назначении ролей ГКУ.

Правила функционирования при назначении ролей ГКУ

1) БС посылает всем узлам СН-пакеты (ClusterHead) для назначения роли каждого ГКУ в отдельности. Каждый такой пакет содержит IDГКУ и список IDподконтрольных ему узлов.
2) Если IDузла, получившего СН-пакет, равен ID ГКУ, то узел назначается в качестве ГКУ.
3) Узел, назначенный в качестве ГКУ начинает рассылку hello-СН-пакета, который также содержит список узлов, подконтрольных данному ГКУ.
4) Если узел, принявший пакет hello-СНвходит в число подконтрольных узлов, указанных в этом пакете, то узел назначается в качестве подконтрольного определенного кластера и в нем заполняется вторая внутренняя структура - ToCHPath (подобная ToBSPath, табл. 3.2), где LTуже инициируется от ГКУ, рассылающего hello-СН-пакет. При этом,power-пакеты не отправляются. Данная структура будет использоваться в дальнейшем только для передачи данных до ГКУ.
5) Если IDузла, получившего СН-пакет, не равен IDГКУ, то производится поиск сопоставления IDГКУ в записях структуры ToBSPath. При наличии совпадений, записи дополняются значением статуса СНравным 1 в третьем поле (см. рис 1).
6) Для пересылки данных будет использоваться две структуры:
    1. Если данные предназначены для БС, то следующий узел-получатель выбирается по структуре ToBSPath по приоритету: наличие статуса СН и наименьшее значение LT. В случае

отсутствия во внутренней структуре узла со статусом CH, будет динамически назначаться ретранслятор – узел с наименьшим LT. При этом, во избежание зацикливания **пакет не может быть послан узлу, от которого он пришел**.

2. Если данные предназначены для ГКУ, в случае если узел является подконтрольным, то данные передаются следующему узлу согласно записям в структуре ToCHPath. Все данные, предназначенные для ГКУ, как только будут им получены, автоматически считаются предназначенными для БС и передаются согласно пп. 6.1.

7) Как только узлу назначается роль ГКУ и подконтрольные узлы инициируются, то ГКУ посылает на БС пакет подтверждения.
8) Получив пакеты подтверждения от всех ГКУ, БС считает, что сеть полностью настроена.

Правила функционирования в обычном режиме

1) Все подконтрольные узлы передают пакеты с данными для ГКУ согласно пп. 6.2 Правил функционирования при назначении ролей ГКУ.
2) Все ГКУ передают пакеты с данными для БС согласно пп. 6.1 Правил функционирования при назначении ролей ГКУ.
3) Если БС меняет свое местоположение таким образом, что перестает видеть какой-либо из ГКУ, то hello-пакеты рассылаются вновь для обновления данных LTво внутренней структуре ToBSPath, но power-пакеты уже не отправляются.
4) Если подконтрольные узлы не могут передать данные на ГКУ, то они начинают посылать fault-пакеты на БС согласно структуре ToBSPath.
5) Если узел «А» не может передать данные на другой узел «Б» в течение времени MaxTransmitExecution, то узел «Б» удаляется из внутренней структуры ToBSPathузла «А».

6) Если БС получила fault-пакеты от более 50% всех ГКУ определенного кластера, тоназначает в данном кластере новый ГКУ и высылает в сеть пакет СН-пакет дляобновления данных этого кластера, который работает в соответствии Правиламифункционирования при назначении ролей ГКУ.

3.3.1 Работа способа нейросетевой кластеризации в составе протокола EDNCP

Способ нейросетевой кластеризации основан на архитектуре ИНС Кохонена, обучаемой по Конструктивному методу.Логика работы данного способа кластеризации при работе в разработанном протоколе маршрутизации EDNCPпредставлена на рис. 3.2. Согласно данному протоколу, информация об уровнях радиовидимости каждого узла относительного соседей собирается базовой станцией, в качестве которой может выступать планшет, ноутбук или сервер. Информация о радиовидимости представляется в виде матрицы радиовидимости, которую описывает представленный граф на рис. 3.1. Непосредственно матричное представление этого графа подается на вход нейронной сети Кохонена, далее производится обучение последней на всем множестве входных данных. Обучение осуществляется в соответствии с Конструктивным методом [64]. Далее, обученная сеть производит кластеризацию на основании тех же входных данных, в результате которой получается многоинтервальная межкластерная структура, на основании которой производится маршрутизацию данных [95, 123].

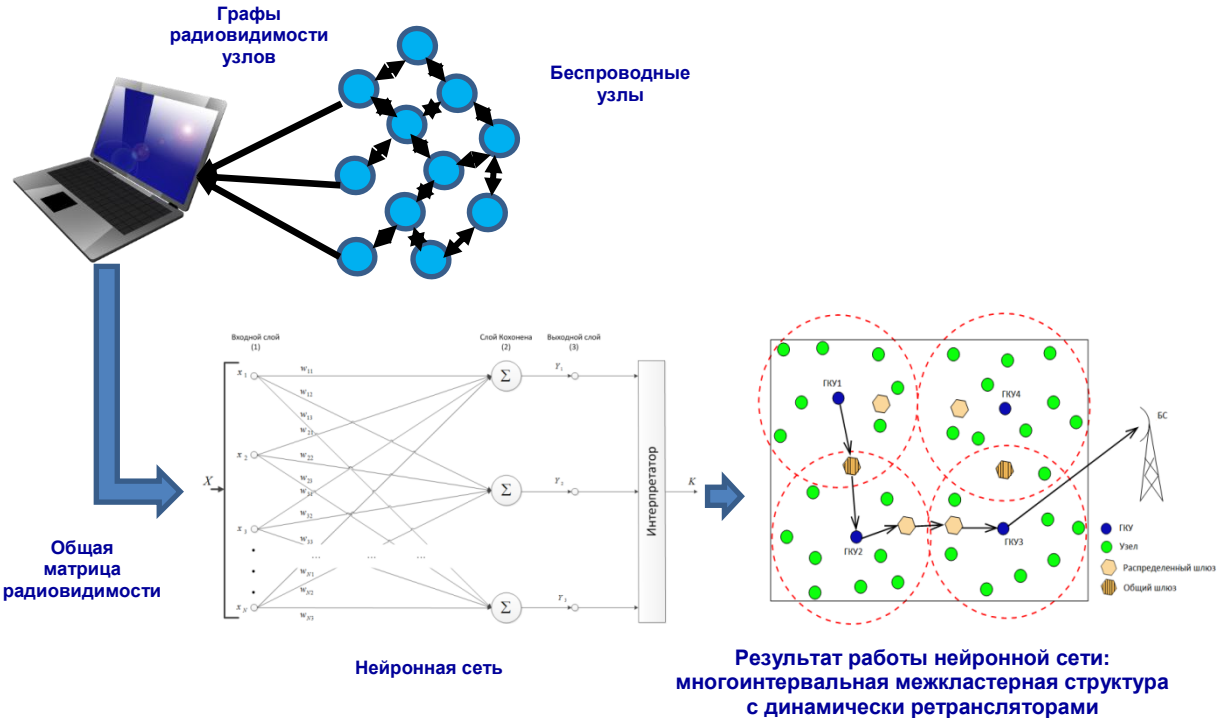


Рисунок 3.2. Схема работы способа нейросетевой кластеризации в составе разработанного протокола маршрутизации

Выводы по главе 3

Разработаны способы кластеризации – нейросетевой и матричный, которые направлены на кластеризацию узлов БСС в иерархических протоколах маршрутизации, повышая их энергоэффективность, выражаемую в продлениижизненного цикла сети [95].

Способы кластеризации основаны на оптимальной иерархической модели связности узлов, которая исследована и выбрана в качестве базовой в главе 2 и использует для кластеризации данные только об энергетической видимости узлов. Это данные об уровне радиовидимостикаждого узла сети по отношению ко всем своим соседним узлам. Такой входной набор параметров для кластеризации является достоверным и призван обеспечить высокую точность при кластеризации узлов сети.

Для способа нейросетевой кластеризации разработан протокол EDNCP, позволяющий наиболее полно раскрыть возможности разработанного способа. Достигаемый технический результат в использовании способа

нейросетевой кластеризации и протокола EDNCP,заключается в автоматическом построении беспроводной сенсорной сети с оптимальными в отношении энергосбережения, путями маршрутизации, благодаря качественному делению БСС на кластеры. Тем самым, техническое решение, основанное на нейронной сети Кохонена [64, 65, 79] должно продлить периоджизненный цикл БСС, а также построить сеть на основании актуальных данных о расположении всех узлов беспроводной сенсорной сети.

ГЛАВА 4. Моделирование способа нейросетевой кластеризации, нейросетевого протокола маршрутизации и сравнение разработанного протокола с известными аналогами

4.1. Моделирование способов кластеризации

С целью подтверждения описанной теории в отношении логики работы разработанных способов кластеризации БСС, их коррекции и проверки адекватности, созданы две программно-моделирующие среды [100, 101]. Данные программы позволяют размещать на 2D-поверхности беспроводные сенсорные узлы с возможностью их желаемого позиционирования как вручную, так и случайным образом [97, 106].

Программы предоставляют возможность автоматического расчета матрицы радиовидимости, которая как было отмечено в главе 3, является математическим описанием БСС для подачи на вход ИНС в разработанных способах кластеризации.

Интерфейс программно-моделирующих сред позволяет задать входные параметры работы разработанных способов кластеризации и на наглядном примере размещения узлов БСС, произвести их кластеризацию с посредством способа нейросетевой кластеризации, либо матричного способа кластеризации, в зависимости от программы [95].

Программы зарегистрированы в официальном государственном реестре программ ЭВМ Российской Федерации, что подтверждают свидетельства:

- Свидетельство о государственной регистрации программы для ЭВМ №2014660980.Махров С.С., Ерохин С.Д. Способ нейросетевой кластеризации беспроводной сенсорной сети [99].
- Свидетельство о государственной регистрации программы для ЭВМ №2014660979.Махров С.С., Ерохин С.Д. Матричный способ кластеризации беспроводной сенсорной сети [98].

4.1.1. Описание интерфейсов программно-моделирующих сред

На рисунке 4.1 представлен интерфейс моделирующей среды для способа нейросетевой кластеризации.

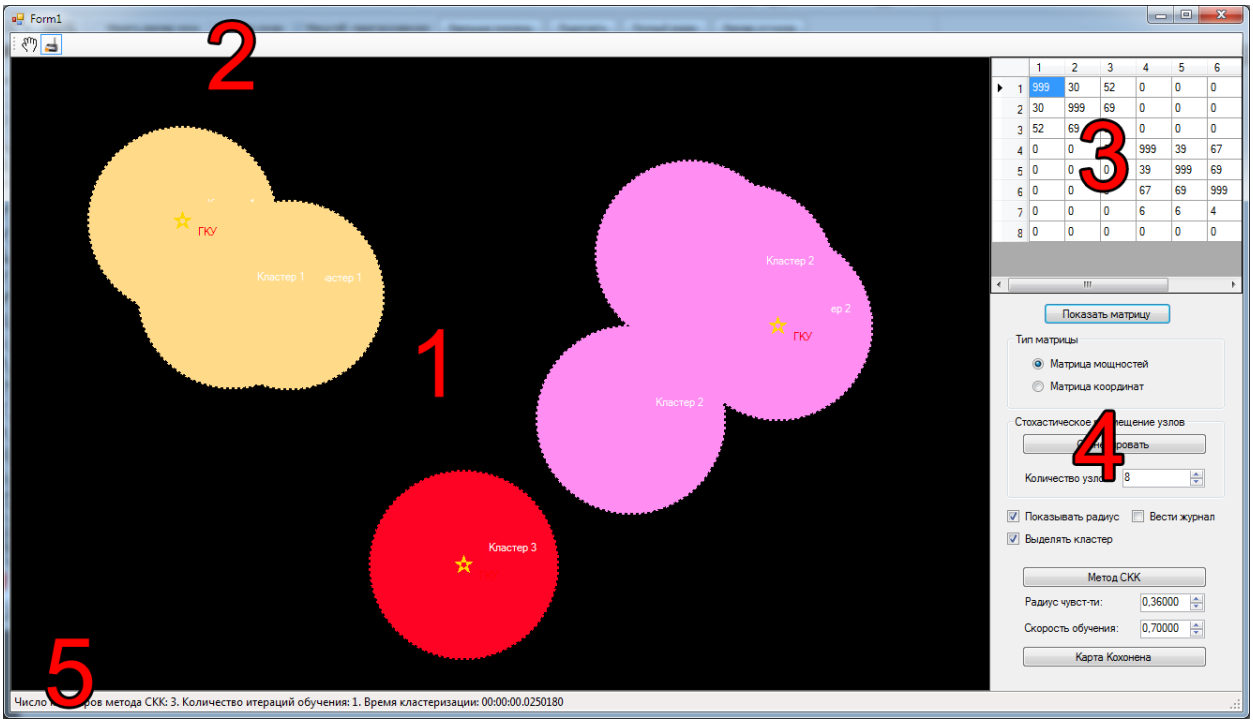


Рисунок 4.1. Программно-моделирующая среда. Способ нейросетевой кластеризации.

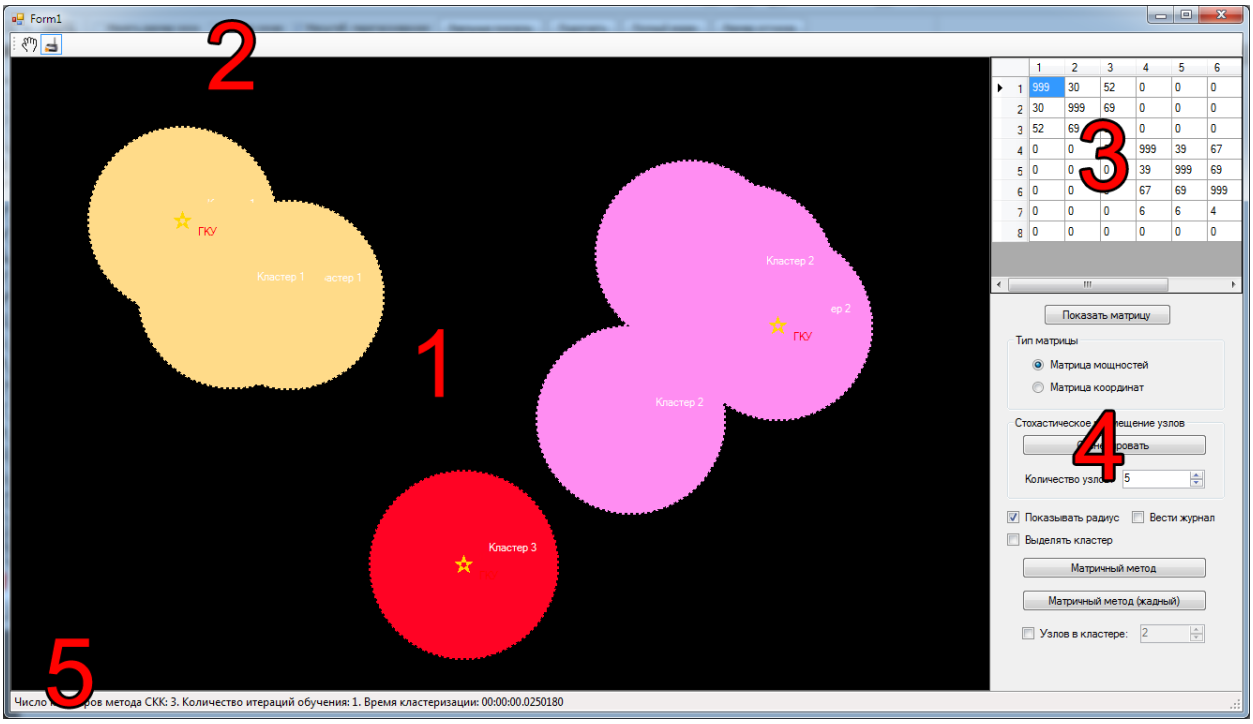
В соответствии с рис. 4.1, цифрами обозначены следующие структурные элементы:

1) Область размещения узлов сети.
2) Панель инструментов для операций над узлами сети на двумерном полотне.
3) Матрица радиовидимости для всех узлов сети.
4) Панель параметров, состоящая из следующих элементов:
    - Выбор типа матрицы: матрица радиовидимости или матрица координат.
    - Стохастическое размещение узлов – задание количества узлов, которые будут случайным образом размещены на двумерном полотне.

- Показывать радиус – отображения радиуса радиовидимости узла.
- Выделять кластер – закрашивание узлов цветами для лучшей видимости их принадлежности к кластерам.
- Вести журнал – ведение журнала операций.
- Радиус чувствительности – параметр ИНС, в соответствии с рекомендациями в главе 3, необходимо задавать для корректной работы в пределах от 0.22 до 0.36.
- скорость обучения – параметр ИНС, в соответствии с рекомендациями в главе 3, необходимо задавать равным 0.7.
- Метод СКК – применение способа нейросетевой кластеризации для размещенных узлов БСС на двумерном полотне.

5) Строка состояния – отображение аналитических результатов кластеризации: время и количество кластеров.

На рисунке 4.2 представлен интерфейс моделирующей среды для матричного способа кластеризации.



В соответствии с рис. 4.2, обозначенные цифрами элементы совпадают с рис. 4.1 за исключением части элементов, обозначенных цифрой 4. Вместо параметров предыдущего способа кластеризации, представлены элементы

управления для инициации запуска матричного способа кластеризации с возможностью задания входного параметра – максимальное количество узлов, которые могут войти в кластер.

4.1.2. Моделирование способа нейросетевой кластеризации

Произведем моделирование способа нейросетевой кластеризации. В качестве входных данных будем задавать количество узлов N, радиус чувствительности нейронов R. Результатами будет визуальная оценка качества кластеризации и времени t, за которое она была произведена.Также на выходе будет известно количество сформированных кластеров K.

Испытание №1.

Входные параметры:

N = 15; R = 0.26.

Результаты:

t = 0.045 секунд; K = 9.Визуализация представлена на рис. 4.3.

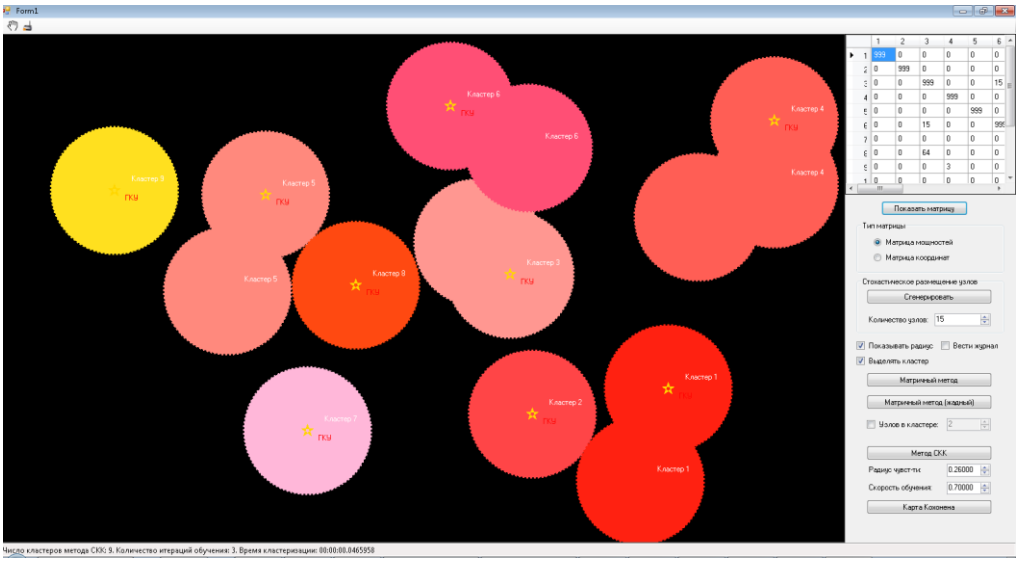


Рисунок 4.3. Испытание №1. Визуализация результатов способа нейросетевой кластеризации.

Вывод: пересекающиеся и близкие друг к другу узлы были объединены в один кластер, изолированные узлы вошли в свой собственный отдельный

кластер, соответственно, множество соседей для таких узлов является пустым $Z \in \varnothing$. ГКУ у кластеров выделены корректно.

Испытание №2.

Входные параметры:

N = 1000; R = 0.36.

Результаты:

t = 4 секунд; K =17. Визуализация представлена на рис. 4.4.

Рисунок 4.4. Испытание №2. Визуализация результатов способа нейросетевой кластеризации.

Вывод: на основании рис. 4.4 видим, что визуально кластеризация произведена правильно, кластеры были выделены, ГКУ определены. На кластеризацию 1000 узлов было затрачено всего 4 секунды.

Испытание №3.

Входные параметры:

N = 1000; R = 0.22.

Результаты:

t = 20минут; K =124. Визуализация представлена на рис. 4.5.

Рисунок 4.5. Испытание №2. Визуализация результатов способа нейросетевой кластеризации.

Вывод: с уменьшением R, увеличивается число кластеров, поскольку радиус охвата для каждого кластера снижается. Следовательно, указание малого Rпозволит формировать малые по размеру кластеры.

Испытание №4.

Входные параметры:N = 100; R = 0.36.

Результаты:t = 0.23 секунды; K =24. Визуализация представлена на рис. 4.6.

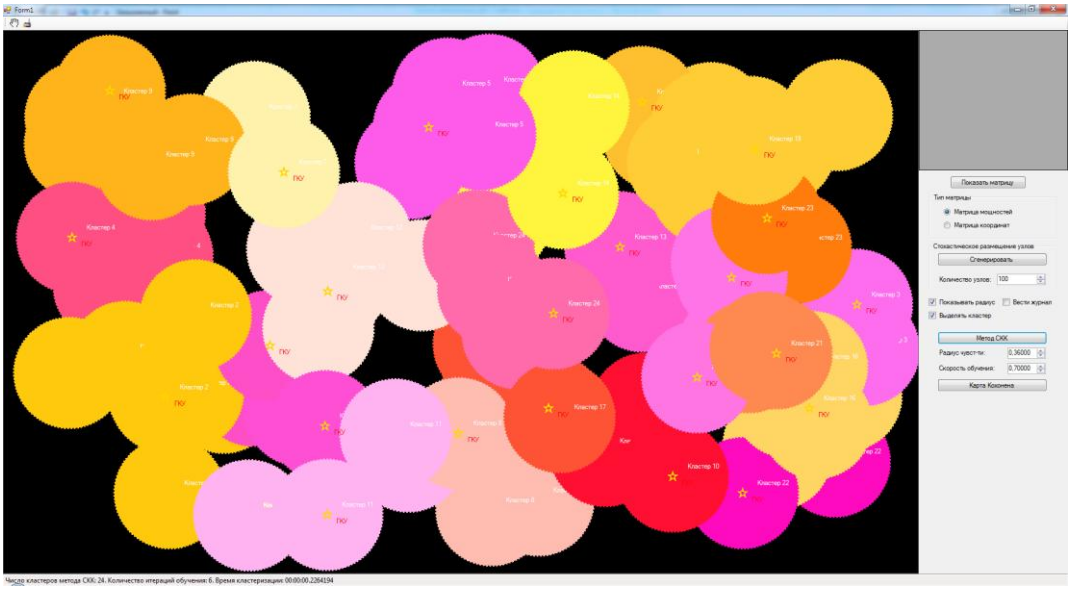


Рисунок 4.6. Испытание №4. Визуализация результатов способа нейросетевой кластеризации.

Вывод: кластеры выделены корректно, изолированных узлов и разорванных кластеров не наблюдается, ГКУ выделены правильно.

Испытание №5.

Входные параметры:

N = 100; R = 0.28.

Результаты:

t = 0.24 секунды; K =43. Визуализация представлена на рис. 4.6.

Рисунок 4.7. Испытание №5. Визуализация результатов способа нейросетевой кластеризации.

Вывод: с уменьшением радиуса чувствительности нейронов, как и в предыдущих испытаниях, радиус охвата каждого кластера уменьшился,и следовательно, в каждый кластер вошло меньшее число узлов.

4.1.3. Моделирование матричного способа кластеризации

Произведем моделирование матричного способа кластеризации. В качестве входных данных будем задавать количество узлов N, количество узлов на кластер N_K. Результатами,как и в предыдущем параграфе, будет визуальная оценка качества кластеризации и времени t, за которое она была произведена. Входные параметры будем задавать аналогично как для способа нейросетевой кластеризации.

Испытание №1.

Входные параметры:

N = 15;

Модификация: без модификаций.

Результаты:

t = 0.049секунд; K =10. Визуализация представлена на рис. 4.8.

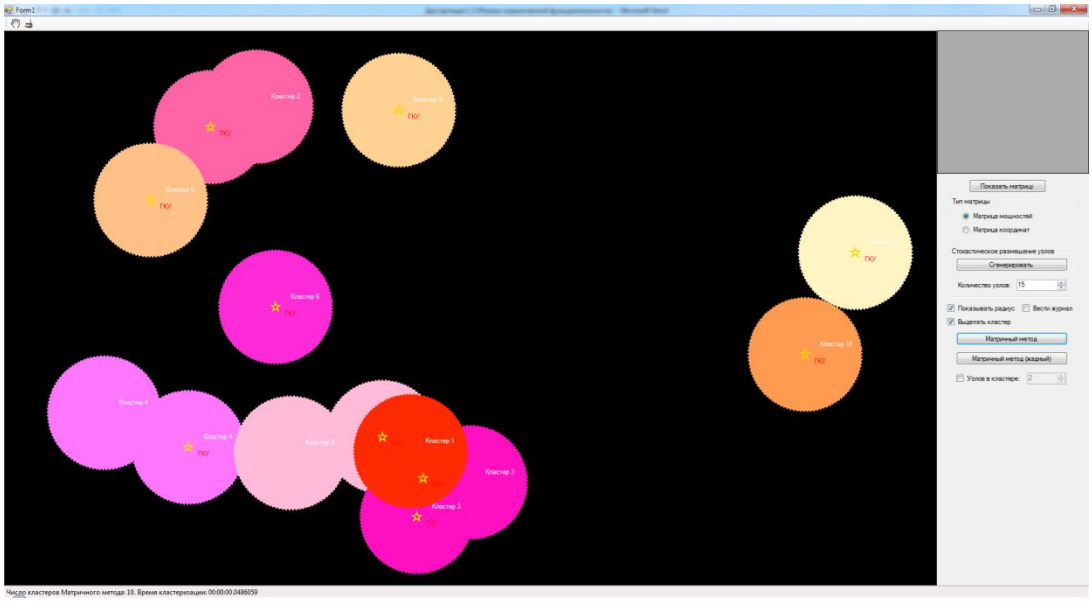


Рисунок 4.8. Испытание №1. Визуализация результатов матричного способа кластеризации.

Вывод: визуально кластеры были выделены правильно. ГКУ в центрах кластеров также были назначены. На осуществление кластеризации было затрачено больше времени по сравнению со способом нейросетевой кластеризации на 0,016 секунд.

Испытание №2.

Входные параметры:

N = 1000;

Модификация: жадный.

Результаты:

t = 43 секунды; K =23. Визуализация представлена на рис. 4.9.

Рисунок 4.9. Испытание №2. Визуализация результатов матричного (жадного) способа кластеризации.

Вывод: визуально кластеры сформированы верно, ГКУ также определены. На осуществление кластеризации было затрачено значительно больше времени по сравнению со способом нейросетевой кластеризации с радиусом чувствительности нейронов R=0.36. Разница составляет: 39секунд.

Испытание №3.

Входные параметры:

N = 1000.

Результаты:

t = 1.11 минуты; K =504. Визуализация представлена на рис. 4.10.

Рисунок 4.10. Испытание №3. Визуализация результатов матричного способа кластеризации.

Вывод: число кластеров получилось достаточно велико, что может повлиять на эффективность маршрутизации данных. Фактически в среднем на один кластер приходится около 2 узлов.Матричный метод отработал быстрее нейросетевого, но выделил кластеры не энергоэффективно, низкая скорость работы способа нейросетевой кластеризации обусловлено слишком низким радиусом чувствительности нейроновR = 0.22, но, тем не менее, кластеры были выделены энерегоэффективно.

Испытание №4.

Входные параметры:

N = 100;

Модификация: жадный.

Результаты:

t = 0.33 секунды; K =15. Визуализация представлена на рис. 4.11.

Рисунок 4.11. Испытание №4. Визуализация результатов матричного (жадного) способа кластеризации.

Вывод: кластеры образованы правильно, разрывов не наблюдается, ГКУ также корректно определены. Время кластеризации больше, чем было

затрачено способом нейросетевой кластеризации на 0.16 секунды при R = 0.36

Испытание №5.

Входные параметры:

N = 100;

Модификация: без модификаций.

Результаты:

t = 0.35 секунды; K =53. Визуализация представлена на рис. 4.12.

Рисунок 4.12. Испытание №5. Визуализация результатов матричного способа кластеризации.

Вывод: кластеры образованы правильно, разрывов не наблюдается, ГКУ также корректно определены. Время кластеризации больше, чем было затрачено способом нейросетевой кластеризации на 0.19 секунды при R = 0.28.

В ходе моделирования также были поставлены эксперименты с другими критериями кластеризации: географические координаты узлов, уровни остаточной энергии. В результате моделирования, сеть давала адекватные результаты кластеризации.

Однако, уровень остаточной энергии не должен использоваться в качестве единственного параметра кластеризации, так как на основании данного критерия невозможно определить как далеко друг от друга находятся узлы и могут ли они быть в радиовидимости друг для друга.

4.1.4. Сравнение нейросетевого и матричного способов кластеризации на основании моделирования

В ходе моделированияв предыдущем параграфе, получены визуальные результаты кластеризации, которые указывают на то, что кластеризация производится адекватно согласно предлагаемым способам. При этом, важно отметить, что для подтверждения репрезентативности данных результатов, дополнительно было произведено также 300 испытаний на каждый способ, и полученные в этих испытаниях наблюдения примерно равны данным, полученным в предыдущем параграфе.

Таким образом, испытания предыдущего параграфа в общем случае подтверждают зависимости, выявленные на этой выборке в случае других экспериментов, на другом множестве значений.На основании произведенных экспериментов предыдущего параграфа, составим следующую сводную таблицу результатов:

Таблица 4.1. Сравнение результатов кластеризации

№	Входные параметры			Выходные параметры		
	N	**R**	**Способ кластеризации**	**t, секунд**	**K**	**Примечание**
1	15	0.26	Нейросетевой	0.045	9	Оба способа отработали корректно, но нейросетевой завершил кластеризацию немного быстрее
		X	Матричный	0.049	10	
2	1000	0.36	Нейросетевой	4	17	Оба способа отработали корректно, но нейросетевой способ завершил кластеризацию значительно быстрее, разница
		X	Матричный(жадный)	43	23	

						составляет 39 секунды
3	1000	0.22	Нейросетевой	1200	124	Матричный метод отработал быстрее, но выделил кластеры не энергоэффективно, низкая скорость работы способа нейросетевой кластеризации обусловленаслишкомнизким радиусом чувствительности нейронов, но, тем не менее, кластеры выделены энерегоэффективно
		X	Матричный	71	504	
4	100	0.36	Нейросетевой	0.23	24	Оба способа отработали корректно, но нейросетевой завершил кластеризацию немного быстрее
		X	Матричный (жадный)	0.33	15	
5	100	0.28	Нейросетевой	0.24	43	
		X	Матричный	0.35	53	

На основании таблицы 4.1. видно, что нейросетевой способ работает быстрее линейного алгоритма, лежащего в основе матричного способа кластеризации. Скорость работы способа нейросетевой кластеризации зависит от радиуса чувствительности нейронов, который, как было рекомендовано в главе 3 необходимо задавать в пределах от 0.22 до 0.36. При задании малого значения R, получим большее число кластеров, и, соответственно в каждый кластер войдет меньшее число узлов. Скорость кластеризации также зависит от R, так при его малых значениях кластеризация осуществляется дольше. Значение Rследует задавать малым при низкой плотности размещения узлов. При высокой плотности, наоборот, значение необходимо задать больше, но в зависимости от желаемого размера кластера. В любом случае, абсолютно верным будет задание среднего значения R, но, тем не менее, необходимо учитывать специфику конкретных узлов и желаемых размеров кластеров.

Матричный способ кластеризации уступает в скорости способу нейросетевой кластеризации, и стандартный вариант в большинстве случаев сильно дробит сеть на кластеры, в результате чего падает энергоэффективность иерархической структуры. «Жадная» модификация данного способа позволяет выделять кластеры энергоэффективно, но также проигрывает по скорости способу нейросетевой кластеризации. Матричный способ кластеризации без модификаций в большинстве случаев на выходе предоставляет такое же количество кластеров, как и способ нейросетевой кластеризации при R = 0.22. Такое же суждение верно для жадной модификации матричного способа и способа нейросетевой кластеризации при Rблизком к значению 0.36.

Нейросетевой способ позволяет приблизительно задать размер кластера, а матричный способ – максимальное количество узлов, которое должно войти в кластер.

Все эксперименты по моделированию разработанных способов кластеризации производились на рабочей станции – персональном компьютере с характеристиками:

- процессор: Core 2 Duo, E8400, 3.0 Ггц;
- оперативная память: 4ГБ;
- операционная система: Windows 7 Professional.

Фактически, для работы моделирующей программы потребовалось 1,5 Гб оперативной памяти.

4.2. Сравнение разработанного протоколаEDNCP с известными аналогами

С целью доказательства эффективности разработанного протокола EDNCPпо сравнению с известными аналогами, выбраны наиболее известные и широко распространенные протоколы, с которыми произведено сравнение: LEACH, TEEN,GAF.

Для сравнения протоколов по данному показателю произведено моделирование, а также анализ сравнение их по известным справочным характеристикам.

4.2.1 Моделирование протоколов маршрутизации

Для оценки эффективности работы протоколов маршрутизации в БСС используется метрикажизненного цикла сети. Оценить данный показатель возможно путем моделирования протоколов маршрутизации.

Моделирование производилось в среде MathWorksMATLABR2014b. Программные коды, задающие логику работы протоколов маршрутизации LEACH, TEEN, GAF[3, 113, 114]в операторах MATLAB (рис. 4.13)взяты с официального сайта MathWorks[115].

```
12      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13      % This is the LEACH [1] code we have used.                              %
14      % The same code can be used for FAIR if m=1                              %
15      % [1] W.R.Heinzelman, A.P.Chandrakasan and H.Balakrishnan,               %
16      %      "An application-specific protocol architecture for wireless      %
17      %       microsensor networks"                                           %
18      %      IEEE Transactions on Wireless Communications, 1(4):660-670,2002  %
19      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20
21 -    clear;
22
23      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24
25      %Field Dimensions - x and y maximum (in meters)
26 -    xm = 100;
27 -    ym = 100;
28      %x and y Coordinates of the Sink
29      %sink.x =0.5 * xm;
30      %sink.y = ym + 50;
31 -    sink.x=50;
32 -    sink.y=175;
33      %sink.x=0.5*xm;
34      %sink.y=0.5*ym;
```

Рисунок 4.13. Вступительная часть к коду симуляции MATLAB, указывающая на использование кода протокола LEACH.

В указанных выше кодах симуляции протоколов в MATLAB принята широко известная классическая модель потребления энергии в БСС, которая была разработана группой исследователей во главе с HeinzelmanW. R.[20, 116, 117]. Данная модель основана на таком наблюдении, что главными потребителями энергии в БСС являются подсистемы передачи данных. В таблице 4.2 приведено краткое описание характеристик данной модели.

Таблица 4.2. Характеристики классической модели потребления энергии в БСС.

Радио режим	**Количество потребления энергии**
Передающая электроника $(E_{Tx-elec})$ Принимающая электроника $(E_{Rx-elec})$ $(E_{Tx-elec} = E_{Rx-elec} = E_{elec})$	50 нДж / бит
Постоянное усиление e_{fs}:	10 пДж / бит / м^2
Бездействие	40 нДж / бит
Сон	0

В качестве входных значений для моделирования каждого протокола маршрутизации зададим следующие начальные условия:

- размер сенсорного поля (для равномерного случайного размещения беспроводных узлов): 100x100 метров;
- координаты базовой станции: x = 50; y = 50;
- количество беспроводных узлов: 100 шт;
- размер передаваемого пакета данных: 4000 бит;
- начальная энергия $E_{initial}$: 2 Дж.

В результате моделирования получили следующие графики отношения количества функционирующих узлов ко времени (рис. 4.14).

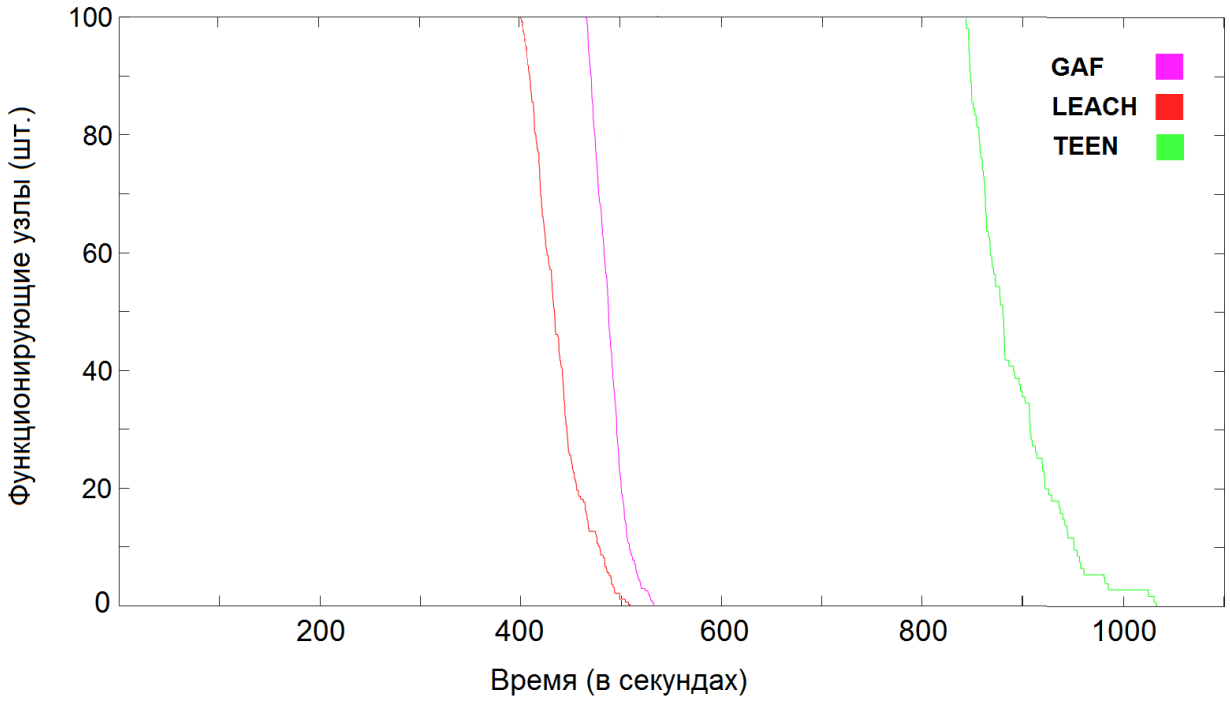


Рисунок 4.14. Графики отношения количества функционирующих узлов ко времени для известных протоколов маршрутизации БСС

Исходя из графиков, выделим для каждого протокола маршрутизации БСС значениепоказателя – жизненного цикла сети, который соответствуют начальной энергии узлов $E_{initial} = 2$ Дж, принятой в качестве входного значения при моделировании. Для наглядности занесем данные в таблицу 4.3.

Таблица 4.3. Полученные значения жизненного цикла сетидля протоколов маршрутизации БСС в результате моделирования.

Протокол маршрутизации БСС	**Жизненный цикл сети (секунд)**
LEACH	512
TEEN	1031
GAF	537

В таблице 4.3 приведены значения жизненного цикла сети под управлением различных протоколов при начальной энергии $E_{initial} = 2$ Дж.

В качестве элемента питания узлов БСС достаточно часто используют 2 элемента питания АА с энергией~ 18878.4 Дж. На основании известного жизненного цикла сети при начальной энергии в 1 Дж, для оценки данных показателей, произведем их вычисление при типовом значенииэнергии источника питания в 18878.4 Дж.Полученные результаты запишем в таблицу 4.4.

Таблица 4.4. Теоретически рассчитанныезначения жизненного цикла сети для ротоколов маршрутизации БСС при типовом значении начальной энергии $E_{initial} = 18878.4$ Дж (2 батарейки АА).

Протокол маршрутизации БСС	**Жизненный цикл сети (дней)**
LEACH	56
TEEN	113
GAF	59

Полученные в результате моделирования и теоретического расчета, значения жизненного цикла сети под управлением различных протоколов БСС, не сильно отличаются от результатов экспериментов других исследователей [118, 119, 120, 121], что указывает на правильность использованных моделей и адекватность проведенного моделирования.

Значения энергии $E_{initial\_m} = 2$ Дж и $E_{initial\_t} = 18878.4$ Дж, использованные при моделировании и теоретическое расчете соответственно, для наглядности переведем в миллиампер-часыпри напряжении питания узлов БСС равным 1.2 В:

$$q = \frac{E}{U} \qquad (4.1),$$

где q – заряд, E – энергия, U– напряжение.

Получим следующие значения зарядов, использованных при моделировании и теоретическом расчете соответственно в миллиампер-часах:

$$q_{initial\_m} = \frac{2\ Дж}{1.2\ B} = 0.6\ Кл = 0.6\ Ас = 0.46\ \text{мАч} \quad (4.2)$$

$$q_{initial\_t} = \frac{18874.4\ Дж}{1.2\ B} = 15728.6\ Кл = 15728.6\ Ас = 4369.05\ \text{мАч} \quad (4.3)$$

Для моделирования жизненного цикла сети, функционирующей под управлением разработанного в данном диссертационном исследовании протоколаEDNCP, произведена доработка описанной и разработанной ранее моделирующей среды, в которой производились исследования в области кластеризации узлов БСС. Моделирующая среда дополнена возможностью учета потребления энергии узлами сети согласно классической энергетической модели потребления энергии [20, 116, 117].

В результате моделирования получили значениежизненного цикла БСС под управлением разработанного протокола EDNCP приприведенных выше начальных условиях, равным 1309 секунд (рис. 4.15).

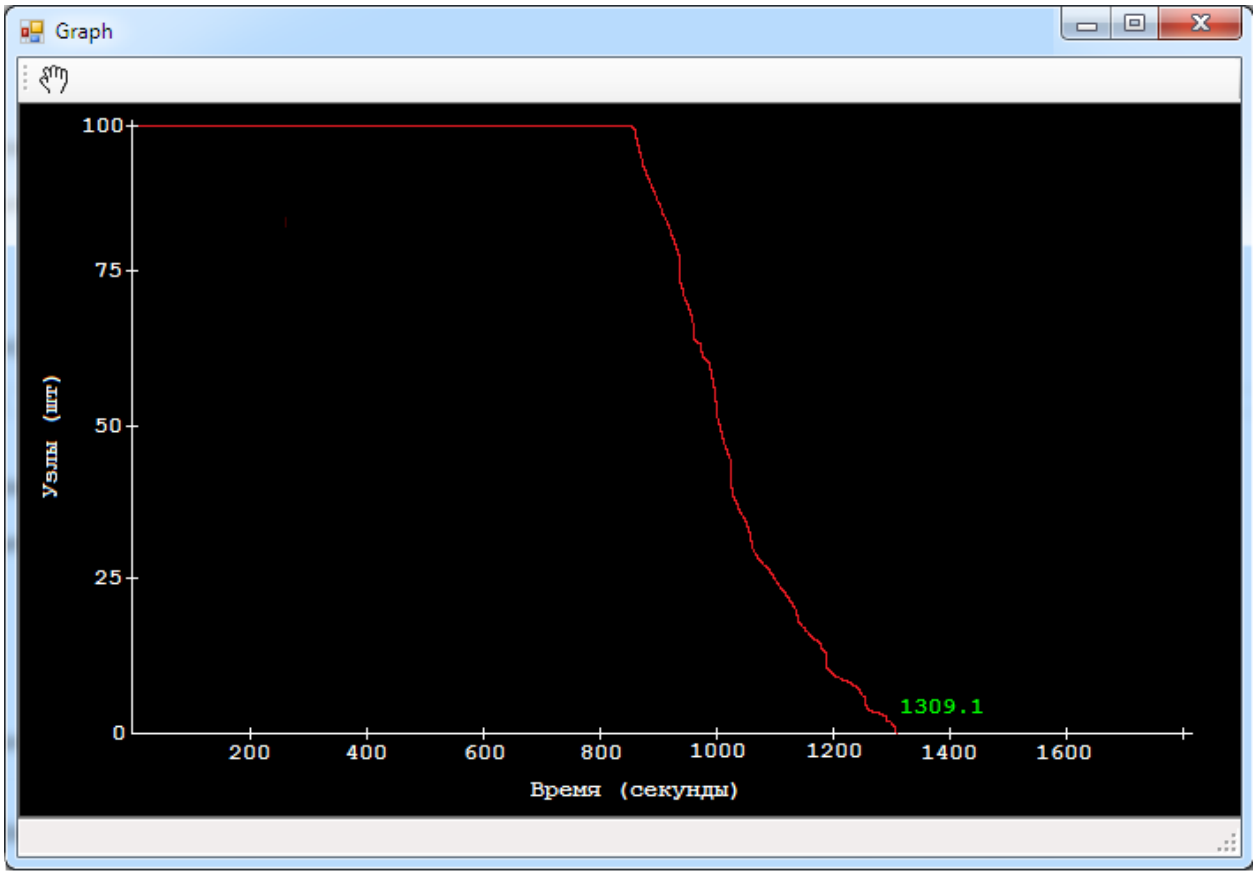


Рисунок 4.15. График отношения количества функционирующих узлов ко времени для протокола EDNCP.

Запишем полученные данные для протокола EDNCPв общие таблицы показателей эффективности (табл. 4.5 и табл. 4.6):

Таблица 4.5. Полученные значения жизненного цикла сети для протоколов маршрутизации БСС, а также протокола EDNCP в результате моделирования.

Протокол маршрутизации БСС	**Жизненный цикл сети (секунд)**
LEACH	512
TEEN	1031
GAF	537
EDNCP	1309

Таблица 4.6. Теоретически рассчитанные значения жизненного цикла сети для протоколов маршрутизации БСС, а также протокола EDNCP при типовом значении начальной энергии $E_{initial} = 18878.4$ Дж (2 батарейки AA).

Протокол маршрутизации БСС	**Жизненный цикл сети (дней)**
LEACH	56
TEEN	113
GAF	59
EDNCP	143

Из всех моделируемых ранее аналогов, наиболее эффективным является TEEN. При сравнении TEENс EDNCPприходим к следующему выводу.

Жизненный цикл БСС под управлениемEDNCPбольше, чем у протокола TEENна 27%.

Следовательно,жизненный цикл БСС под управлениемEDNCPбольше известных аналогов на27%.

4.2.2. Сравнение справочных характеристик известных протоколов маршрутизации с EDNCP

Для анализа эффективностихарактеристик разработанного протокола энергетических расстояний нейросетевой маршрутизации (EDNCP), произведено сравнение со справочными характеристиками указанных ранее известных протоколов маршрутизации. Ниже приведена таблица, отражающая результаты сравнения (см. табл. 4.7).

Таблица 4.7. Сравнение справочных характеристик наиболее известных протоколов маршрутизации БСС с протоколом EDNCP

Критерий/Протокол		EDNCP	LEACH	TEEN	GAF
масштабируемость (кол-во узлов)		свыше 10 000	~100	~1000	~1000
независимость от модулей определения местоположения		Да	Да	Нет	Нет
мобильная базовая станция		Да	Нет	Нет	Нет
резервные пути маршрутизации		Да	Нет	Нет	Нет
Возможность внутрикластерной передачи данных		Да	Нет	Нет	Да
Возможность межкластерной передачи данных		Да	Нет	Да	Нет
Сложность реализации		Высокая	Средняя	Средняя	Средняя
Качество кластеризации	БС имеет данные обо всех узлах, входящих в состав сети	Да	Нет	Нет	Нет
	Наличие узлов одного кластера внутри другого	Нет	Да	Нет	Нет
	Наличие изолированных узлов, не способных передавать данные на БС	Нет	Нет	Нет	Нет
	Возможность задания размера кластера	Да	Нет	Нет	Нет

По результатам сравнительного анализа, выявлено, что разработанный протокол эффективнее, по отношению к остальным, использованным в сравнении. Недостатком является лишь сложность его реализации.

EDNCPпревосходит данные протоколы по таким характеристикам как:

- высокая масштабируемость – свыше 10 000 узлов;
- независимость работы протокола от модулей определения местоположения (GPS/ГЛОНАС);
- возможность доставки пакетов от ГКУ до БС через узлы-посредники – другие ГКУ или ретрансляторы;
- возможность использования мобильной базовой станции;
- возможность динамического назначения ретрансляторов, что исключает зависимость от определенных узлов, заранее назначенных в качестве ретрансляторов статически;
- точность формирования кластеров.

Перечисленные выше достоинства указывают на эффективность использования протокола EDNCP по сравнению с современными, наиболее известными протоколами маршрутизации БСС.

Выводы по главе 4

В данной главе произведено компьютерное моделирование работы [111] Способа нейросетевой кластеризации, Матричного способа кластеризации БСС и Протокола энергетических расстояний нейросетевой кластеризации (EDNCP).

Моделирование способов кластеризации осуществлялось в собственных программно-моделирующих средах, написанной на языке программирования высокого уровня – C# в среде разработки программных решений MicrosoftVisualStudio 2010. Исходный код программного обеспечения представлен в Приложении 5 и Приложении 6.

На основании компьютерного моделирования разработанных способов кластеризации, выявлено, что способ нейросетевой кластеризации показал свою высокую эффективность на различном множестве входных данных. В общем случае, как было отмечено ранее, испытания были произведены на 300 различных входных наборах данных, включающих в себя: размещение узлов БСС с различной плотностью, различные значения R, количества узлов N.

Таким образом, на выходе способа нейросетевой кластеризации можно получить энергоэффективную иерархическую кластерную структуру, которая может быть использована в иерархических протоколах маршрутизации данных БСС.

Произведено моделирование в среде MATLABнаиболее известных протоколов маршрутизации и их сравнение с разработанным протоколом маршрутизации EDNCPпо метрике жизненного цикла сети. Для сравнения были выбраны протоколы: LEACH, TEEN, GAF[3, 113, 114]. В результате, выявлено, что разработанный протокол EDNCPэффективнее известных аналогов на 27%.

Выполнен анализ эффективности характеристик разработанного протокола EDNCP с известными справочными характеристиками указанных

ранее протоколов. Выявлено, что разработанный протокол маршрутизации EDNCP обладает более эффективными параметрами, чем известные аналоги.

На основании произведенного моделирования и сравнительного анализа, выявлено, что разработанный протокол EDNCPпозволяет формировать кластеры точнее, чем известные аналоги.

ЗАКЛЮЧЕНИЕ

В диссертации решен комплекс теоретических и практических проблем самоорганизации БСС и маршрутизации данных с использованием механизмов искусственного интеллекта на базе нейронных сетей. Основные результаты работы заключаются в следующем:

1. Предложено использовать ИНС для кластеризации БСС. Благодаря механизмам искусственного интеллекта стало возможным при кластеризации учитывать множество параметров, которые описывают каждый узел. Например, уровень радиовидимости узла по отношению ко всем своим соседям, находящимся в потенциальной доступности для передачи данных. ИНС решают задачу многопараметрической оптимизации на всем множестве входных данных, описывающих каждый узел БСС.
2. Произведен анализ моделей связности узлов в беспроводной сенсорной сети, который позволил выбрать энергоэффективную архитектуру для проектироемого протокола, в основу которого положен способ кластеризации узлов БСС. Многоинтервальная иерерхическая модель связности является наиболее эффективной в отношении энергосбережения, благодаря агрегации данных с узлов на ГКУ, образующих вокруг себя кластер. При этом, ГКУ передает данные опосредованно через другие ГКУ преимущественно или через релеи на БС.
3. Предложена матрица радиовидимости, являющаяся математическим описанием связности узлов сети и радиовидимости каждого узла по отношению ко всем остальным узлам сети. Данное представление узлов позволяет формализовать данные обо всех узлах БСС, и представить их в соответствующим виде для подачи на вход ИНС.

Имея данные обо все узлах сети, возможно наиболее оптимально выделить кластеры в сети, благодаря полноте информации.

4. Исследована эффективность кластеризации с помощью нейронной сети – Самоорганизующейся карты Кохонена, обучаемой по Конструктивному методу. На основании данного исследования, выявлено, что для кластеризации БСС адекватные результаты можно получить при использовании автоматически обучаемых ИНС (обучаемые «без учителя»). Произведено анализ существующих, наиболее известных архитектур ИНС, на основании чего выявлено, что наиболее эффективно использовать ИНС Кохонена, благодаря отсутствию недостатков, присущих другим архитектурам ИНС. Существующие недостатки ИНС Кохонена возможно устранить посредством нестандартного метода обучения – Конструктивного метода.
5. Разработан способ нейросетевой кластеризации беспроводной сенсорной сети, основанный на архитектуре ИНС Кохонена, обучаемой по Конструктивному методу. Эффективность и адекватность способа кластеризации подтверждена компьютерным моделированием.
6. Разработан матричный способ кластеризации беспроводной сенсорной сети.Эффективность и адекватность способа кластеризации подтверждена компьютерным моделированием
7. Разработан протокол энергетических расстояний нейросетевой кластеризациина основе способа нейросетевой кластеризации, которыйпозволяет повысить жизненный цикл сети на 27% по сравнению с существующими протоколами маршрутизации данных БСС.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

АРТ – адаптивнаярезонанснаятеория.

АЦП – аналого-цифровой преобразователь.

БС – базоваястанция.

БСС– беспроводная сенсорная сеть.

ГКУ– главный кластерный узел.

ИВ – Интернетвещей.

ИНС – искусственная нейронная сеть.

МР– матрица (мощностей) радиовидимости.

ОС - операционная система.

СКК – самоорганизующаяся карта Кохонена.

СУ – сенсорный узел.

ЦПУ – центральное процессорное устройство.

ANN – artificialneuralnetwork.

ART –adaptive resonance theory.

BS – basestation.

CH – clusterhead.

CSMA-CA – carrier sense multiple access with collission avoidance.

CWTA - winner takes all with conscience.

EDNCP–energydistanceneuralclusteringprotocol.

IoT – Internet of Things.

LT – life time of packet.

PM – powermatrix

QoS – quality of service.

RSS – received signal strength(мощностьполучаемогосигнала).

TDMA – time division multiple access.

WSN – wireless sensor network.

WTM – winner takes all.

СЛОВАРЬ ТЕРМИНОВ

Базовая станция – управляющий узел всей сети, на который передаются данные со всех остальных узлов.

Гетерогенная сеть – сеть, состоящая из узлов разного типа, например с разными по емкости источниками энергии.

Главный кластерный узел – это узел, на который передаются данные со всех подконтрольных ему узлов кластера.

Гомогенная сеть – сеть с однородной структурой, в которой используется совместимое оборудование и общие протоколы обмена.

Жизненный цикл беспроводной сенсорной сети – интервал времени между началом функционирования и гибелью последнего из функционирующих сенсорных узлов.

Интернет вещей –концепция, согласно которой планируется практически каждое устройство оснастить подключением к сети Интернет.

Искусственная нейронная сеть – математическая модель, а также её программная или аппаратная реализация, построенная по принципу организации и функционирования биологических нейронных сетей

Кластеризация– разделение узлов сети на отдельные группы, во главе которых ставится узел, производящий сбор данных со всех узлов кластера. Такой узел называется главным кластерным узлом (ГКУ). ГКУ передает данные на БС станцию напрямую, либо через других ГКУ.

Матрица радиовидимости – квадратная матрица размера Nна N, которая содержит уровни радиовидимости в процентах всех узлов сети относительно друг друга. N – количество узлов сети.

Сенсорное поле – пространство, которое покрывается узлами сенсорной сети.

Центроид – центральный узел, по отношению ко всем остальным узлам в кластере.

СПИСОК ЛИТЕРАТУРЫ

1. Abbasi A.A., Younis M. A survey on clustering algorithms for wireless sensor networks // Computer Communications. - 2007. - №30. - Pp. 2826-2841.

2. Adeel A., Abid A.M., Sohail J. Energy Aware Intra Cluster Routing for WSN// International Journal of Hybrid Information Technology. - 2010. - Vol.3, №1. -Pp. 29-48.

3. Al-Karaki J. and Kamal A. E. Routing Techniques in Wireless Sensor Networks: A Survey// IEEE Communications Magazine. –2004. – Vol.11, №6. - Pp. 6-11.

4. Applications of Wireless Sensor Networks in Next Generation Networks: technical paper // International Telecommunication Union. ITU-T; developed by Valery Butenko, Anatoly Nazarenko, Viliam Sarian, Nikolay Sushchenko and Aleksandr Lutoshkin. – [Geneva], 2014.

5. Bornhovd C., Lin T., Haller S., and Schaper J. Integrating smart items with business processes: An experience report // IEEE, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICCS). - Hawaii: IEEE Computer Society, 2005. - Pp. 227-230.

6. Boukerche A., Cheng X., and Linus J. Energy-aware data-centric routing in microsensor networks // ACM MSWiM, Proceedings ACM MSWiM, in conjunction with ACM MobiCom. - San Diego, CA: ACM MSWiM, 2003. - Pp. 42-49.

7. Braginsky D., Estrin D. Rumor routing algorithm in sensor networks // Proceedings ACM WSNA, in conjunction with ACM MobiCom'02. - Atlanta, GA: ACM WSNA, 2002. - Pp. 22-29.

8. Buttyan L., Schaffer P. PANEL: Position-based Aggregator Node Election in Wireless Sensor Networks // IEEE, Proceedings of IEEE Inetrnational Conference on Mobile Adhoc and Sensor System. - Pisa: IEEE, 2007. - Pp. 1-9.

9. Chang W., Cao G, La Porta T. Dynamic proxy tree-based data dissemination schemes for wireless sensor networks // IEEE, Proceedings IEEE MASS'04. - Fort Lauderdale, FL: IEEE, 2004. - Pp. 21-30.

10. Chaves L., S´a de Souza L., Muller J., Anke J. Service lifecycle management infrastructure for smart items // MidSens, Proceedings of the international workshop on Middleware for sensor networks (MidSens). - Melbourne, Australia: MidSens, 2006. - Pp. 25-30.

11. Chi C., Hatler M. Industrial wireless sensor networking. Technical report // ON World. - 2004. - №4. - Pp. 46-51.

12. Chu M., Haussecker H., Zhao F. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks // International Journal of High Performance Computing APplications. - 2002. - vol.16. - №3. - Pp. 293-313.

13. Dargie W., Poellabauer C. Fundamentals of wireless sensor networks: theory and practice. John Wiley and Sons, 2010. –p. 330.

14. Du X., Lin F. Improving routing in sensor networks with heterogeneous sensor nodes // IEEE, Proceedings IEEE VTC'05. - Dallas, TX: IEEE, 2005. - Pp. 2528-2532.

15. Fukushima K., Miyake S., Takayuki I. Neocognitron: A neural network model for a mechanism of visual pattern recognition // IEEE Transaction on Systems, Man and Cybernetics. - 1983. - №13(5). - Pp. 826–834.

16. GPS (GlobalPositioningSystem): [Электронныйресурс] // Санкт-Петербургскийтехникумгеодезииикартографии. - 2006.-Режим доступа: http://www.spbtgik.ru/book/6220.htm (Дата обращения: 14.04.2014).

17. Karl H., Willig A. Protocols and Architectures for Wireless Sensor Networks. John Wiley & Sons, 2005. -p. 245.

18. Heinzelman W. R., Kulik J., Balakrishnan H. Adaptive protocols for information dissemination in wireless sensor networks // ACM, Proceedings ACM MobiCom '99. - Seattle, WA: ACM, 1999. - Pp. 174-185.

19. HeinzelmanW.R., Chandrakasan A., Balakrishnan H. An Application-Specific Protocol Architecture for Wireless Microsensor Networks // IEEE, Transactions on Wireless Communications (October 2002) vol. 1(4). - Seattle, WA: IEEE, 2002. - Pp. 660-670.

20. Heinzelman W.R., Chandrakasan A., Balakrishnan H. Energy-efficient Communication Protocol for Wireless Microsensor Networks, vol. 8 // IEEE, Proceedings of the Thirty Third Hawaii International Conference on System Sciences (HICSS '00). - Washington, DC, USA: IEEE Computer Society, 2000. - C. 8020.

21. Hussein M.S. Survey of Routing Protocols in Wireless Sensor Networks. International Journal of Sensors and Sensor Networks // International Journal of Sensors and Sensor Networks. - 2014. - vol. 2. - №1. - Pp. 11-16.

22. Ibrahiem M. M., El E., Ramakrishnan S. Wireless Sensor Networks: From Theory to Applications. CRC Press, 2013. - p. 799.

23. Intanagonwiwat C., Govindan R., Estrin D Directed diffusion: A scalable and robust communication paradigm for sensor networks // ACM MobiCom, Proceedings ACM MobiCom'00. - Boston, MA: ACM MobiCom, 2000. - Pp. 56-67.

24. Intanagonwiwat C., Govindan R., Estrin D., Heidemann J., Silva F. Directed diffusion for wireless sensor networking // IEEE/ACM Transactions on Networking. - 2003. - vol. 11. - №1. - Pp. 2-16.

25. Johnson D. B.et al. “Dynamic Source Routing in Ad Hoc Wireless Networks”, in Mobile Computing, edited by Tomas Imielinski and Hank Korth, Kluwer Academic Publishers, ISBN: 0792396979, 1996, Chapter 5, Pp. 153-181.

26. Karp B., Kung H.T. GPSR: Greedy perimeter stateless routing for wireless networks // ACM, Proceedings ACM MobiCom'00. - Boston, MA: ACM, 2000. - Pp. 243-254.

27. Kulik J., HeinzelmanW., Balakrishnan H. Negotiation-based protocols for disseminating information in wireless sensor networks // Wireless Networks. - 2002. - vol. 8. - №2/3. - Pp. 169-185.

28. Leen G., Heffernan D. Vehicles without wires // Computing and Control Engineering Journal. - 2001. - №12 (5). - Pp. 205–211.

29. Li L., Halpern J.Y. Minimum-energy mobile wireless networks revisited // IEEE, Proceedings IEEE ICC’01. - Helsinki, Finland: IEEE, 2001. - Pp. 278-283.

30. Lindsey S., Raghavendra C.S. PEGASIS: Power-efficient Gathering in Sensor Information System // IEEE, Proceedings IEEE Aerospace Conference. - vol. 3. - Big Sky. - MT: IEEE, 2002. - Pp. 1125-1130.

31. Lindsey S., Raghavendra C.S., Sivalingam K. M. Data gathering algorithms in sensor networks using energy metrics // IEEE, Transactions on Parallel and Distributed Systems. - 2002. - vol. 13. - №9. - Pp. 924-935.

32. Lindsey S., Raghavendra C.S., Sivalingam K. M. Data gathering in sensor networks using the energy delay metric // Proceedings IPDPS'01. - San Francisco, CA: IPDPS, 2001. - Pp. 2001-2008.

33. Lou W. Data gathering in sensor networks using the energy delay metric // IEEE, Proceedings of IEEE MASS’05. - Washington DC: IEEE, 2005. - Pp. 1-8.

34. Low A. Evolution of Wireless Sensor Networks for Industrial Control // Technology Innovation Management Review. - Ottawa, Canada: Carleton University, 2013. - Pp. 5-12.

35. Sustainable Wireless Sensor Networks / Maimour M., Zeghilet H., Francis L., Winston Seah, Yen Kheng Tan.InTech (Ed.), 2010. – P. 584.

36. Makhrov S.S. Prospects of Nanotechnologies Development in Automated Control Systems // Second Forum of Young Researchers. In the framework of International Forum «Education Quality – 2010». - Izhevsk: Publishing House of ISTU, 2010. - Pp. 370-375.

37. Manjeshwar A., Agrawal D. P. TEEN: A Protocol for Enhanced Efficiency in Wireless Sensor Networks // Proceedings of the 1st International Workshop on

Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing. - San Francisco, CA: 2001. - Pp. 2009-2015.

38. Manjeshwar A., Agrawal D. P. APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks // Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing. - San Francisco, CA: 2001. - Pp. 1009-1015.

39. McCulloch W.S., Pitts W. A logical Calculus of Ideas Immanent in Nervous Activity // Bull. Mathematical Biophysics. - 1943. - Vol. 5. - Pp. 115-133.

40. Nath B., Niculescu D. Routing on a curve // ACM SIGCOMM Computer Communication Review. - 2003. - Vol. 33. - №1. - Pp. 155-160.

41. Nissan E. Computer Applications for Handling Legal Evidence, Police Investigation and Case Argumentation. Springer, 2012. - Pp. 1428.

42. Ossama Y., Fahmy S. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-efficient Approach // IEEE, Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. - West Lafayette, IN, USA: IEEE, 2001. - Pp. 366-379.

43. Ossama Y., Marwan K., Srinivasan R. Node Clustering in Wireless Sensor Networks: Recent Developments and Deployment Challenges // IEEE, IEEE Network. - Tucson, AZ, USA: IEEE, 2006. - Pp. 20-25.

44. Rodoplu V., Meng T. H. Minimum energy mobile wireless networks // IEEE Journal on Selected Areas in Communications. - 1999. - Vol. 17. - №8. - Pp. 1333-1344.

45. Sadagopan N., Krishnamachari B., Helmy A. The ACQUIRE mechanism for efficient querying in sensor networks // Proceedings SNPA'03. - Anchorage, AK: SNPA, 2003. - Pp. 149-155.

46. Salami A.F., Anwar F., Priantoro A.U. An Investigation into Clustering Routing Protocols for WSN // Sensors and Transducers Journal. - 2009. – Vol. 105. - Issue 6. - Pp. 2-5.

47. Schaap H. Wireless sensor network standard for logistic processes: Master's thesis - Enschede, 2007. -P. 145-166

48. Senouci M.R., Melouk A., Senouci H., Aissani A. Performance evaluation of network lifetime // Journal of Network and Computer Applications. - 2012. - №35 (4). - Pp. 1317-1328.

49. Singh S. K., Singh M. P. and Singh D. K. Routing Protocols in Wireless Sensor Networks – A Survey // International Journal of Computer Science & Engineering Survey (IJCSES). - 2010. - №Vol. 1. - №2. - Pp. 63-83.

50. Senouci M. R., Mellouk A., Senouci H., Aissani A. Performance evaluation of network lifetime spatial-temporal distribution for WSN routing protocols // Journal of Network and Computer Applications. - 2012. - Vol. 35. - Issue 4. - Pp. 1317-1328.

51. Sushruta M., Alok R., Abhishek K., Vishal C., Preksha V., Lalit B. Study of Cluster Based Routing Protocols in Wireless Sensor Networks // International Journal of Scientific & Engineering Research. - 2012. - №Vol. 3. - Issue 7. - Pp. 21-37.

52. Wireless Sensor Network (WSN): Architectural Design issues and Challenges. Ajay Jangra et al. // (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 09, 2010, Pp. 3089-3094.

53. Xing G., Lu C., Pless R., Huang Q. On greedy geographic routing algorithms in sensing-covered networks // ACM, Proceedings ACM MobiHoc'04. - Tokyo, Japan: 2004. - Pp. 31-42.

54. Xiu Y., Heidemann J. and Estrin D. Geography-informed energy conservation for ad-hoc routing // IEEE, Proceedings ACM/IEEE MobiCom'01. - Rome, Italy: 2001. - Pp. 70-84.

55. Y.2069. Terms and definitions for the Internet of things: ITU-T Recommendation. – approved 07/2012. – Geneva: International Telecommunication Union, 2012, Pp. 4.

56. Y.2222. Sensor Control Network and related applications in Next Generation Network environment: ITU-T Recommendation. – approved 04/2013. – Geneva: International Telecommunication Union, 2013, Pp. 30.

57. Yao Y., Gehrke J. The Cougar approach to in-network query processing in sensor networks // SGIMOD Record. - 2002. - Vol. 31. - №3. - Pp. 9-18.

58. Ossama Y., Fahmy S. Heed: A hybrid, Energy-efficient, Distributed Clustering Approach for Ad-hoc Networks // IEEE Transactions on Mobile Computing. - 2004. - Vol. 3. - №4. - Pp. 366-379.

59. Yu Y., Govindan R., Estrin D. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks // Technical Report. UCLA Computer Science Department. - 2001. - UCLA/CSD-TR-01-0023. - Pp. 1-11.

60. Zorzi M., Rao R. R. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Mutlihop performance // IEEE Transactions on Mobile Computing. - 2003. - №Vol. 2. - №4. - Pp. 337-348.

61. Аджемов А.С. Теоретические границы и возможности их достижения в будущих инфокоммуникациях // Электросвязь. - 2003. - № 11. - С.15–18.

62. Баскаков С. С. Беспроводные сенсорные сети: вопросы и ответы // Автоматизация в промышленности. - 2008. - №4. - С. 34–35.

63. Баскаков С. С., Оганов В. И. Беспроводные сенсорные сети на базе платформы MeshLogic // Электронные компоненты. - 2006. - №8. - С. 65–69.

64. Баталов А.С Конструктивный метод обучения нейронной сети Кохонена // Сборник научных трудов SWorld. Материалы международной научно-практической конференции «Перспективные инновации в науке, образовании, производстве и транспорте 2012». - Выпуск 2. Том 1. - Одесса: КУПРИЕНКО, 2012. - С. 94-99.

65. Баталов А.С. Методы повышения эффективности обучения нейронной сети Кохонена // Вестник Пермского университета. - Сер.: Математика. Механика. Информатика. - 2012. - Вып. 3 (11). - С. 86-93.

66. Бохан К.А., Федоренко Н.И. Сравнительный анализ видов нейронных сетей для обработки мультимедиа данных // Радиіоелектронні і комп'ютерні системи. - 2008. - №6 (33). - С. 298-306.

67. Вишневский В.М., Гайкович Г.Ф. Беспроводные сенсорные сети в системахпромышленной автоматики // Электроника. - 2008. - №1. - С. 106-110.

68. Воронцов К. В. Лекции по искусственным нейронным сетям от 21 декабря 2007г: [Электронный ресурс] // Федеральное государственное бюджетное учреждение науки. Вычислительный центр им. А.А. Дородницына Российской академии наук. - 2014. - Режим доступа: http://www.ccas.ru/voron/download/NeuralNets.pdf (Дата обращения: 14.04.2014).

69. Восков Л.С. Беспроводные сенсорные сети и прикладные проекты. Автоматизация и ИТ в энергетике // Отраслевой научно-производственный журнал. - 2009. - №2-3. - С. 44-49.

70. Восков Л.С., Ефремов С.Г. К вопросу о времени автономной работы сенсорных сетей // Качество. Инновации. Образование. - 2012. - №7. - С. 61-67.

71. Восков Л.С., Комаров М.М. Позиционирования датчиков беспроводной сенсорной сети как способ энергосбережения // Датчики и системы. - 2012. - № 1. - С. 34-38.

72. Восков Л.С., Курпатов Р.О. Сравнительный анализ методов локализации в беспроводных сенсорных сетях // Качество. Инновации. Образование. - 2011. - № 3. - С. 35-39.

73. Горбаченко В.И. Сети и карты Кохонена: [Электронный ресурс] // Научно-исследовательский центр самоорганизации и развития систем. - 2010. -Режим доступа: http://gorbachenko.self-organization.ru/index.html (Дата обращения: 01.02.2014).

74. Дмитриев А.С., Кузьмин Л.В., Юркин В.Ю. Сверхширокополосные беспроводные сенсорные сети на основе хаотических радиоимпульсов // Изв. вузов. Прикладная нелинейная динамика. - 2009. - Том 17, №4. - С. 90–104.

75. Ежов А.А., Шумский С.А. Нейрокомпьютинг и его применения в экономике и бизнесе: учебник / под ред. проф. Харитонова В.В. - М.: Изд. МИФИ, 1998. – 224 с.

76. Зиновьев А. Ю. Визуализация многомерных данных. - Красноярск: Изд. Красноярского государственного технического университета, 2000. - 180 с.

77. Каллан Р. Основные концепции нейронных сетей: Пер. с англ. - М.: Издательский дом «Вильямс», 2001. - 291 с.

78. Кальченко Д. Нейронные сети: на пороге будущего // КомпьютерПресс. - 2005. - №1. - С. 86-90.

79. Кохонен Т. Самоорганизующиеся карты. Пер. 3-го англ. изд. - М.: БИНОМ. Лаборатория знаний. 2008. - 655 с.: с ил. ISBN 978-5-94774-352-4.

80. Кучерявый А.Е., Салим А. Выбор головного узла кластера в однородной беспроводной сенсорной сети // Электросвязь. - 2009. - №8. - С. 32-36.

81. Кучерявый А.Е. Интернет вещей // Электросвязь. - 2013. - №1. - С. 21-24.

82. КучерявыйА.Е., ПрокопьевА.В., КучерявыйВ.А. Саморганизующиесясети. – СПб.: Любавич, 2011.

83. Кучерявый Е.А. Беспроводные сенсорные сети и их роль в прогрессивном обществеXXI века // Первый профессиональный журнал для специалистов в области телекоммуникаций и информационных технологий "Информационные телекоммуникационные сети". - 2006. - № 2. - С. 36-45.

84. Кучерявый Е.А., Молчан С.А., Кондратьев В.В. Принципы построения сенсоров и сенсорных сетей // Электросвязь. - 2006. - № 6. - С. 10-15.

85. Кучерявый Е.А., Салим А. Выбор головных узлов в однородной беспроводной сенсорной сети для обеспечения полного покрытия // 64-я Научно – техническая конференция, посвященная Дню Радио. - СПб: Изд-во СПбГЭТУ "ЛЭТИ", 2009. - С. 45-51.

86. КучерявыйЕ.А., Салим А. Диаграммы Вороного для беспроводных сенсорных сетей// 64-я Научно – техническая конференция, посвященная Дню Радио. - СПб: Изд-во СПбГЭТУ "ЛЭТИ", 2009. - С. 67-74.

87. Манжула В.Г., Федяшов Д.С. Нейронные сети Кохонена и нечеткие нейронные сети в интеллектуальном анализе данных // Научный журнал «Фундаментальные исследования». - 2011. - №4. - С. 108-113.

88. Махров С.С. Автоматическое построение беспроводной сенсорной сети на основе искусственной нейронной сети //Мобильные телекоммуникации. – 2014, №6-7 (134). -С. 45-47.

89. Махров С.С. Анализ архитектур самообучающихся нейронных сетей в задаче кластеризации узлов беспроводной сенсорной сети // Мобильные телекоммуникации. - 2014. - №4-5 (133). С. 68-71.

90. Махров С.С. Беспроводные сенсорные сети в военно-тактических задачах // Техника средств связи: научно-технический сборник. Выпуск 2 (141). - СПб: Изд-во Политехн. ун-та, 2013. - С. 176-179.

91. Махров С.С. Возможности применения нейросетевых технологий в беспроводных сенсорных сетях // Перспективные технологии в средствах передачи информации: Материалы 10-ой международной научно-технической конференции / Владим. гос. университет; редкол.: А.Г. Самойлов (и др). - Владимир: ВлГУ. - Том 1. - 2013. - С.108-112.

92. Махров С.С. Ерохин С.Д. Особенности и ограничения архитектур операционных систем беспроводных сенсорных сетей при разработке технологических решений // Труды Северо-Кавказского филиала Московского технического университета связи и информатики. - Ростов-на-Дону: ПЦ «Университет» СКФ МТУСИ. - 2013. -С. 142-144.

93. Махров С.С. Использование систем моделирования беспроводных сенсорных сетей NS-2 и OMNET++ // T-COMM: Телекоммуникации и транспорт. - 2013. - №10. - С. 67-69.

94. Махров С.С. Исследование связности узлов в иерархических протоколах беспроводных сенсорных сетей // Фундаментальные проблемы радиоэлектронного приборостроения / Материалы Международной научно- технической конференции «INTERMATIC-2013», 2–6 декабря 2013 г., Москва. / Под ред. академика РАН А. С. Сигова. – М.: Энергоатомиздат, 2013. - Часть 4. – С. 186-189.

95. Махров С.С. Нейросетевая кластеризация узлов беспроводной сенсорной сети // T-COMM: Телекоммуникации и транспорт. - 2014. - №6. - С. 31-35.

96. Махров С.С. Перспективы внедрения беспроводных сенсорных сетей для обеспечения экономических и бизнес-процессов // Мобильные телекоммуникации. - 2013. - №3 (123). - С. 47-49.

97. Махров С.С. Программный комплекс для формирования функциональных геометрических моделей «ПИКАР // XI научно-практическая конференция «Дни науки - 2011. Ядерно-промышленный комплекс Урала»: Том 2. Тезисы докладов.- Озерск: ОТИ НИЯУ МИФИ, ФГУП «ПО МАЯК», 2011. -С. 38-40.

98. Махров С.С., Ерохин С.Д. Свидетельство о государственной регистрации программы для ЭВМ «Матричный способ кластеризации беспроводной сенсорной сети» №2014660979 от 21.10.2014, правообладатели: Махров С.С., Ерохин С.Д.

99. Махров С.С., Ерохин С.Д. Свидетельство о государственной регистрации программы для ЭВМ «Способ нейросетевой кластеризации беспроводной сенсорной сети» №2014660980 от 21.10.2014, правообладатели: Махров С.С., Ерохин С.Д.

100. Махров С.С., Мишарин Д.А., Аввакумов В.Д. Моделлер геометрических моделей и чертежей объектов PiBuilder // Сборник трудов IVконференции «Автоматизация и прогрессивные технологии в атомной отрасли» (АПТ-2009), Том II. - Новоуральск: изд. НГТИ, 2009. - С. 119-120.

101. Махров С.С., Николаев Н.А. Библиотека подпрограмм для построения графиков ColdGraphX// Сборник научных трудов. Международная научно-практическая конференция «Снежинск и наука – 2009. Современные проблемы атомной науки и техники». - Снежинск: изд. СГФТА, 2009. - С. 279-281.

102. НейроПроект. Аналитические технологии для прогнозирования и анализа данных: [Электронный документ].-Режима доступа: http://www.neuroproject.ru/tutorial.php. (Дата обращения: 20.04.2013).

103. Постарнак Д.В. Критический анализ моделей нейронных сетей. Вестник Тюменского государственного университета // Вестник Тюменского государственного университета. - 2012. - №4. - С. 162-167.

104. Прошлое, настоящее и будущее Интернета вещей / В.К. Сарьян, Н.А. Сущенко, И.А. Дубнов, Ю.А. Дубнов, С.В. Сахно, А.С. Лутохин // Труды НИИР.-2014. - № 1. - С. 1-7.

105. Салим А. Разработка алгоритмов выбора головного узла в кластерных беспроводных сенсорных сетях: дисс. канд. техн. наук. - Спб., 2010. – 106 с.

106. Сергиевский М.В., Сыроежкин С.Н. Беспроводные сенсорные сети. Часть 3. Средства программирования // КомпьютерПресс. - 2008. - №8. - С. 127-129.

107. Трифонова С. В., Холодов Я. А. Исследование и оптимизация работы беспроводной сенсорной сети на основе протокола ZigBee // Компьютерные исследования и моделирование. - 2012. - № 4. - Том 4.- С. 855-869.

108. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. 2-е изд, испр. / Пер. с англ. М.: ООО «И. Д. Вильямс», 2006. - 356 с.

109. Хайкин С. Нейронные сети: полный курс, 2-е издание: Пер. с англ. - М.: Издательский дом "Вильямс", 2006. - 1104 с.

110. Чубукова И.А. Data Mining.Интернет-университет информационных технологий. -М.: Бином. Лаборатория знаний, 2000.- 326 с.

111. Шелухин О.И., Тенякшев А.М., Осин А.В. Моделирование информационных систем. - М.: Радиотехника, 2005. - 368 с.

112. Юркин В.Ю., Мохсени Т. И. Иерархические подходы к самоорганизации в беспроводных сверхширокополосных сенсорных сетях на основе хаотических радиоимпульсов // Труды МФТИ. - 2012. - № 3. -Том 4. - С. 151-161.

113. Bhattacharyya D., Kim T., Pal S. A Comparative Study of Wireless Sensor Networks and Their Routing Protocols // Sensors. – 2010. №3. - pp. 1506-1523.

114. Liu X. A Survey on Clustering Routing Protocols in Wireless Sensor Networks // Sensors, - 2012. №5. -pp. 1113-1153.

115. MATLABCentral. FileExchange: [Электронный документ].-Режим доступа: http://www.mathworks.com/matlabcentral/fileexchange/. (Дата обращения:11.02.2015).

116. Torres M.G.Energy consumption in wireless sensor networks using GSP: Master's thesis - Medellín, Pittsburgh, 2006. -P. 145-166

117. Liaw J., Chang L., Chu H. Improving lifetime in heterogeneous wireless sensor networks with the energy-efficient grouping protocol // International Journal of Innovative Computing, Information and Control. – 2012. vol. 9, №9. - pp. 6037-6047.

118. Huynh T.T., Hong C.S. Prolonging Network Lifetime via Intra-Cluster Routingin Wireless Sensor Networks // Osaka University, Proceedings Second International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2005). - Osaka, Japan: 2005. - Pp. 130-135.

119. Tan N.D., Han L, Viet N.D.,Jo M. An Improved LEACH Routing Protocol for Energy-Efficiency of Wireless Sensor Networks // Smart Computing Review. - 2012. Vol. 2, №5. - pp. 360-369.

120. Sirdeshpande N., Udupi V. Lifetime Maximization Using Modified Leach Protocol for Energy Efficient Routing In Wireless Sensor Networks // The International Journal Of Engineering And Science. - 2013. - Vol.2, №2. pp. 17-23.

121. Cui X. Research and Improvement of LEACH Protocol in Wireless Sensor Networks // IEEE, IEEE 2007 International Symposium on Microwave, Antenna, Propagation, and EMC Technologies For Wireless Communications. - Hangzhou: 2007. - pp. 251-254

122. РозенблаттФ. Принципынейродинамики: перцептроныитеориямеханизмовмозга = PrinciplesofNeurodynamic: perceptronsandthetheoryofbrainmechanisms. — М.: «Мир», 1965. - 478 с.

123. Махров С.С. Симбиоз беспроводных сенсорных технологий и искусственного интеллекта нейронных сетей // Вестник связи. - 2015. - № 2. - С. 37-39.

Приложение 1. Актвнедрениярезультатовкандидатскойдиссертационнойработыв ПравительствеМосквы

ПРАВИТЕЛЬСТВО МОСКВЫ

ДЕПАРТАМЕНТ ГОРОДСКОГО ИМУЩЕСТВА ГОРОДА МОСКВЫ

Улица Бахрушина, д. 20, Москва, 115054
Телефон: 8 (495) 959-1888, факс: 8 (495) 959-1982
E-mail: dgi@mos.ru, http://www.dgi.mos.ru
ОКПО 16412348, ОГРН 1037739510423,
ИНН/КПП 7705031674/770501001

13.11.2014 № ДГИ-Б-89321/14

на № — от —

АКТ
о внедрении результатов
кандидатской диссертационной работы

Настоящий акт удостоверяет, что результаты диссертационных исследований Махрова Станислава Станиславовича были использованы в практической деятельности органа исполнительной власти Правительства Москвы – Департамента городского имущества города Москвы.

Реализация методов кластеризации нейронных сетей Самоорганизующейся карты Кохонена была использована при проектировании автоматизированной информационной системы документооборота, статистики, анализа и учета показателей качества – «Единая система регистрации документов Департамента городского имущества города Москвы» (ЕСРД ДГИ).

Заместитель руководителя
Департамента городского имущества города Москвы,
кандидат экономических наук Д.Н. Тетушкин

Приложение 2. Актвнедрения в учебный процесс МТУСИ

«УТВЕРЖДАЮ»

Ректор МТУСИ
д.т.н, проф.

_____ Аджемов А.С.

«20» января 2015 г.

АКТ

об использовании результатов кандидатской диссертации

«Использование нейронных механизмов искусственного интеллекта для кластеризации узлов и маршрутизации данных в беспроводных сенсорных сетях»

Махрова Станислава Станиславовича

в учебном процессе кафедры информационной безопасности и автоматизации (ИБиА) ФГОБУ ВПО МТУСИ

Комиссия в составе:

- председателя заведующего кафедрой ИБиА д.т.н., проф. О.И.Шелухина
- членов комиссии:
  С.Д.Ерохина, к.т.н., доцент, доцент кафедры ИБиА
  А.А,Андрюков, к.т.н., доцент, доцент кафедры ИБиА

составили настоящий акт о том, что следующие результаты диссертационной работы С.С.Махрова, полученные им лично, использованы в лекционном курсе и при постановке практических и лабораторных занятий на дисциплине «Сетевые технологии», а именно:

1) Методы кластеризации узлов беспроводной сенсорной сети;
2) Анализ алгоритмов и протоколов маршрутизации;
3) Использование нейронных механизмов для кластеризации узлов БСС.

Председатель комиссии:

_____ /О.И. Шелухин/

Члены комиссии:

_____ /С.Д. Ерохин/

_____ /А.А.Андрюков/

Приложение 3. Свидетельство о государственной регистрации программы для ЭВМ. Способ нейросетевой кластеризации беспроводной сенсорной сети

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014660980

Способ нейросетевой кластеризации беспроводной сенсорной сети

Правообладатели: ***Махров Станислав Станиславович (RU), Ерохин Сергей Дмитриевич (RU)***

Авторы: ***Махров Станислав Станиславович (RU), Ерохин Сергей Дмитриевич (RU)***

Заявка № **2014618745**
Дата поступления **29 августа 2014 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***21 октября 2014 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Приложение 4. Свидетельство о государственной регистрации программы для ЭВМ. Матричный способ кластеризации беспроводной сенсорной сети

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014660979

Матричный способ кластеризации беспроводной сенсорной сети

Правообладатели: ***Махров Станислав Станиславович (RU), Ерохин Сергей Дмитриевич (RU)***

Авторы: ***Махров Станислав Станиславович (RU), Ерохин Сергей Дмитриевич (RU)***

Заявка № **2014618744**
Дата поступления **29 августа 2014 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***21 октября 2014 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Приложение 5. Исходный код программы для ЭВМ. Способ нейросетевой кластеризации беспроводной сенсорной сети

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace NeuroSens
{
    public class Vector
    {
        public double[] InputVector;
        public int BMUIndex = -1;

        public Vector(int count)
        {
            InputVector = new double[count];
        }
    }

    public class VectorsSet
    {
        public List<Vector> vectors;

        public VectorsSet(int ComponentsCount, int VectorCount)
        {
            vectors = new List<Vector>();
            for (int i = 0; i < VectorCount; i++)
            {
                Vector vector = new Vector(ComponentsCount);
                vectors.Add(vector);
            }
        }
    }

    public class NeuroNet
    {
        public int inputs = 0;
        public List<Neuron> neurons;

        public NeuroNet(int inputs_, int neurons_)
        {
            neurons = new List<Neuron>();
            inputs = inputs_;
            Random rand = new Random();
            for (int i = 0; i < neurons_; i++)
            {
                Neuron neuron = new Neuron(i, inputs_, rand);
                neurons.Add(neuron);
            }
        }

        public int AddNeuron()
        {
            Random rand = new Random();
            Neuron neuron = new Neuron(neurons.Count, inputs, rand);
            neurons.Add(neuron);
```

```
            return neurons.Count - 1;
        }
    }

    public class Neuron
    {
        public int number = 0;
        public List<double> weights;

        public Neuron(int number_, int inputs_, Random rand)
        {
            weights = new List<double>();
            number = number_;
            for (int i = 0; i < inputs_; i++)
            {
                weights.Add(rand.NextDouble());
            }

        }
    }

    class SOM
    {
        public List<int> GetClusterHeads(List<List<int>> Clusters, NeuroNet
net, VectorsSet RealVectorSet)
        {
            List<int> ClusterHeads = new List<int>();

            for (int i = 0; i < Clusters.Count; i++)
            {
                List<double> ClusterDist = new List<double>();
                if (Clusters[i].Count > 0)
                {
                    int CH = Clusters[i][0];
                    for (int j = 0; j < Clusters[i].Count; j++)
                    {
                        if (EuclideanDistance(net.neurons[i],
RealVectorSet.vectors[CH].InputVector) > EuclideanDistance(net.neurons[i],
RealVectorSet.vectors[Clusters[i][j]].InputVector))
                            CH = Clusters[i][j];
                        ClusterDist.Add(EuclideanDistance(net.neurons[i],
RealVectorSet.vectors[Clusters[i][j]].InputVector));
                    }
                    ClusterHeads.Add(CH);
                }
            }

            return ClusterHeads;
        }

        public List<int> GetClusterHeads2(List<List<int>> Clusters, int[,]
Matrix)
        {
            int N = Convert.ToInt32(Math.Sqrt(Matrix.Length));
            List<int> ClusterHeads = new List<int>();

            for (int i = 0; i < Clusters.Count; i++) // переборкластеров
            {
                if (Clusters[i].Count > 0)
                {
                    double MaxAverage = 0;
                    int MaxAverageNode = Clusters[i][0];
```

```
                    for (int j = 0; j < Clusters[i].Count; j++) //
переборузловвкластере
                    {
                        double Average = 0;
                        for (int k = 0; k < N; k++) // переборкомпонентовузла
                        {
                            Average += Matrix[Clusters[i][j], k];
                        }
                        if (Average > MaxAverage)
                        {
                            MaxAverage = Average;
                            MaxAverageNode = Clusters[i][j];
                        }
                    }
                    ClusterHeads.Add(MaxAverageNode);
                }
            }

            return ClusterHeads;
        }

        public List<List<int>> GetClusters(int[,] Matrix, int[,] PowerMatrix,
ref int IterNum, ref List<int> ClusterHeads, ref NeuroNet net, bool DoLog,
double Radius, double Speed)
        {
            List<List<int>> Clusters = new List<List<int>>();
            int N = Matrix.GetLength(0);
            int M = Matrix.GetLength(1);

            ProgressForm PF = new ProgressForm();
            PF.Show();
            PF.ProgressPersent = 0;
            PF.ProgressText = "Подготовкаккластеризацииузлов...";
float Sum = 0;

            int ClustersCount = 0;

            int inputNeurons = M;
            int outputNeurons = 1;
            VectorsSet RealVectorSet = new VectorsSet(M, N);
            NormalizeMatrixToVectors(Matrix, ref RealVectorSet);

            net = new NeuroNet(inputNeurons, outputNeurons);
            PF.ProgressPersent = 50;
            PF.ProgressWindowHeight = PF.Height + RealVectorSet.vectors.Count
* 14 * 2;
            IterNum = Study(ref net, RealVectorSet, ref PF, DoLog, Radius,
Speed);

            List<int>  row;
            for (int i = 0; i < net.neurons.Count; i++)
            {
                row = new List<int>();
                Clusters.Add(row);
            }

            for (int i = 0; i < RealVectorSet.vectors.Count; i++)
            {
                int CurrentCluster = Test(net,
RealVectorSet.vectors[i].InputVector);
                Clusters[CurrentCluster].Add(i);
            }
```

```
            CorrectClusters(ref Clusters);

            //ClusterHeads = GetClusterHeads2(Clusters, Matrix);
            ClusterHeads = GetClusterHeads(Clusters, net, RealVectorSet);

            PF.ProgressPersent = 100;
            PF.ProgressText = "Кластеризациязавершена";
            PF.Hide();
            PF.Dispose();
            return Clusters;
        }

        public void CorrectClusters(ref List<List<int>> Clusters)
        {
            int Count = Clusters.Count;
            for (int i = 0; i < Clusters.Count; i++)
            {
                if (Clusters[i].Count == 0)
                {
                    Clusters.RemoveAt(i);
                    CorrectClusters(ref Clusters);
                    break;
                }
            }
        }

        private void FillByRandomVectors(ref VectorsSet EdVectorSet,
VectorsSet RealVectorSet)
        {
            Random Rand = new Random();
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                EdVectorSet.vectors[i] = RealVectorSet.vectors[Rand.Next(0,
RealVectorSet.vectors.Count - 1)];
            }
        }

        private void SetOrderNumbersToVectors(ref VectorsSet EdVectorSet)
        {
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                EdVectorSet.vectors[i].BMUIndex = 0;
            }
        }

        private double GetMaxValue(int[,] Matrix)
        {
            int N = Matrix.GetLength(0);
            int M = Matrix.GetLength(1);

            double MaxValue = Matrix[0, 0];

            for (int i = 0; i < N ; i++)
            {
                for (int j = 0; j < M; j++)
                {
                    if (Matrix[i, j] > MaxValue)
                        MaxValue = Matrix[i, j];
                }
            }
```

```
            return MaxValue;
        }

        private double GetMinValue(int[,] Matrix)
        {
            int N = Matrix.GetLength(0);
            int M = Matrix.GetLength(1);

            double MinValue = Matrix[0, 0];

            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < M; j++)
                {
                    if (Matrix[i, j] < MinValue)
                        MinValue = Matrix[i, j];
                }
            }

            return MinValue;
        }

        public int GetClustersNumber(int[,] Matrix)
        {
            int N = Convert.ToInt32(Math.Sqrt(Matrix.Length));

            int[] P = new int[N];

            for (int i = 0; i < N; i++)
            {
                int temp_p = 0;
                for (int j = 0; j < N; j++)
                {
                    if (Matrix[i, j] != 0)
                        temp_p++;
                }
                P[i] = temp_p;
            }

            int P_network = Mean(P, N);
            int Kopt = Convert.ToInt32(Math.Ceiling((double)N /
(double)P_network));
            return Kopt;
        }

        protected int Mean(int[] Vector, int N)
        {
            int Sum = 0;
            for (int i = 0; i < N; i++)
            {
                Sum += Vector[i];
            }
            int Mean = Convert.ToInt32(Math.Ceiling((double)Sum /
(double)N));
            if (Mean == 0)
                Mean = 1;
            return Mean;
        }

        private int Test(NeuroNet net, double[] InputVector)
        {
            double MinDistance = System.Double.PositiveInfinity;
```

```
            int BMUIndex = -1;
            for (int i = 0; i < net.neurons.Count; i++)
            {
                double tmp_ED = EuclideanDistance(net.neurons[i],
InputVector);
                if (tmp_ED < MinDistance)
                {
                    BMUIndex = i;
                    MinDistance = tmp_ED;
                }
            }
            return BMUIndex;
        }

        private void Record(string Text, bool DoLog)
        {
            if (!DoLog)
                return;
            string appendText = Text + " ";
            File.AppendAllText("log.txt", appendText, Encoding.UTF8);
        }

        private void RecordN(string Text, bool DoLog)
        {
            if (!DoLog)
                return;
            string appendText = Environment.NewLine + Text + " ";
            File.AppendAllText("log.txt", appendText, Encoding.UTF8);
        }

        private void WriteVectorsInfo(VectorsSet EdVectorSet, bool DoLog)
        {
            if (!DoLog)
                return;
            RecordN("Входныевекторы:", DoLog);
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                RecordN("Вектор " + i + ":", DoLog);
                for (int g = 0; g <
EdVectorSet.vectors[i].InputVector.Length; g++)
                {
                    Record(EdVectorSet.vectors[i].InputVector[g] + "  ",
DoLog);
                }
            }
        }

        private void WriteNeuronsInfo(NeuroNet net, bool DoLog)
        {
            if (!DoLog)
                return;
            RecordN("Весанейронов:", DoLog);
            for (int i = 0; i < net.neurons.Count; i++)
            {
                RecordN("Нейрон " + i + ":", DoLog);
                for (int g = 0; g < net.neurons[i].weights.Count; g++)
                {
                    Record(net.neurons[i].weights[g] + "  ", DoLog);
                }
            }
        }
```

```
        private double Median(VectorsSet EdVectorSet)
        {
            double Median = 0;

            int Count = 0;
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                for (int j = 0; j <
EdVectorSet.vectors[i].InputVector.Length; j++)
                {
                    if (EdVectorSet.vectors[i].InputVector[j] == 1)
                        Count++;
                }
            }
            Median = (double)Count / (double)EdVectorSet.vectors.Count;

            return Median;

        }

        private int Study(ref NeuroNet net, VectorsSet EdVectorSet, ref
ProgressForm PF, bool DoLog, double R, double n0)
        {
            File.Delete("log.txt");
            WriteVectorsInfo(EdVectorSet, DoLog);
            WriteNeuronsInfo(net, DoLog);

            double[] EuclideanDistanceSum = new
double[EdVectorSet.vectors.Count];
            EuclideanDistanceSum =
InitializeEuclideanDistanceSum(EuclideanDistanceSum, net, EdVectorSet);
            int IterNum = 0;
            double N = net.inputs;
            double sr = Median(EdVectorSet);
            double Z1 = Math.Sqrt(N) * R + (N / 100) * sr / 200;
            double Z = Math.Sqrt(N) * R /1.3;
            do
            {
                RecordN("", DoLog);
                RecordN("Итерация " + IterNum, DoLog);
PF.ProgressText = "Количество итераций: " + IterNum;
double normfunc = normLearningRate(IterNum, n0);

                for (int k = 0; k < EdVectorSet.vectors.Count; k++) // цикл,
вкоторомпредъявляемсетивходныевектора - InputVector
{
                    RecordN("Обучающий вектор " + k + ":", DoLog);
double MinDistance = System.Double.PositiveInfinity;
                    int BMUIndex = -1;
                    if (IterNum == 0 && k == 0)
                        BMUIndex = net.neurons[0].number;
                    else
                    {
                        for (int i = 0; i < net.neurons.Count; i++)
                        {
                            double tmp_ED = EuclideanDistance(net.neurons[i],
EdVectorSet.vectors[k].InputVector); //находимЕвклидоворасстояниемежду i-
ымнейрономи k-ымвходнымвектором
RecordN("Евклидово расстояние между " + k + "-ым вектором и " + i + "-ым
нейроном = " + tmp_ED, DoLog);
                            if (tmp_ED < MinDistance) // если Евклидово
расстояние минимально, то это нейрон-победитель
```

```
                            {
                                BMUIndex = net.neurons[i].number; // индекс
нейрона-победителя
MinDistance = tmp_ED;
                            }
                        }
                    }

                    if (IterNum == 0 && k == 0)
                    {
                        for (int g = 0; g <
EdVectorSet.vectors[k].InputVector.Length; g++)
                        {
                            net.neurons[BMUIndex].weights[g] =
EdVectorSet.vectors[k].InputVector[g];
                        }
                        continue;
                    }

                    if (MinDistance > Z)
                    {
                        BMUIndex = net.AddNeuron();
                        for (int g = 0; g <
EdVectorSet.vectors[k].InputVector.Length; g++)
                        {
                            net.neurons[BMUIndex].weights[g] =
EdVectorSet.vectors[k].InputVector[g];
                        }
                        continue;
                    }

                    RecordN("", DoLog);
                    RecordN("Нейронпобедитель: " + BMUIndex, DoLog);
                    for (int g = 0; g <
EdVectorSet.vectors[k].InputVector.Length; g++)
                    {
                        net.neurons[BMUIndex].weights[g] += normfunc *
(EdVectorSet.vectors[k].InputVector[g] - net.neurons[BMUIndex].weights[g]);
                    }
                    WriteVectorsInfo(EdVectorSet, DoLog);
                    WriteNeuronsInfo(net, DoLog);
                }
                IterNum++;
            }
            while (!Err(net, EdVectorSet, IterNum, ref PF) && Changes(ref
EuclideanDistanceSum, net, EdVectorSet, IterNum));
            return IterNum;
        }

        private void NormalizeMatrixToVectors(int[,] Matrix, ref VectorsSet
EdVectorSet)
        {
            int N = Matrix.GetLength(0);
            int M = Matrix.GetLength(1);

            if (N == M)
            {
                for (int i = 0; i < N; i++)
                {
                    for (int j = 0; j < M; j++)
                    {
                        if (Matrix[i, j] != 0)
```

```
                            EdVectorSet.vectors[i].InputVector[j] = 1;
                        else
                            EdVectorSet.vectors[i].InputVector[j] = 0;
                    }
                }
            }
            else
            {
                double MaxValue = GetMaxValue(Matrix);
                double MinValue = GetMinValue(Matrix);
                for (int i = 0; i < N; i++)
                    for (int j = 0; j < M; j++)
                        EdVectorSet.vectors[i].InputVector[j] =
(double)(Matrix[i, j] - MinValue) / (double)(MaxValue - MinValue);
            }
        }

        private double normLearningRate(int k, double n0)
        {
            return n0 * Math.Exp((double)-k / 1000.0);
        }

        private double EuclideanDistance(Neuron neuron, double[] InputVector)
        {
            double Sum = 0;
            for (int i = 0; i < InputVector.Length; i++)
            {
                Sum += Sqr(InputVector[i] - neuron.weights[i]);
            }
            return Math.Sqrt(Sum);
        }

        private double EuclideanDistance(Neuron neuron1, Neuron neuron2)
        {
            double Sum = 0;
            for (int i = 0; i < neuron1.weights.Count; i++)
            {
                Sum += Sqr(neuron1.weights[i] - neuron2.weights[i]);
            }
            return Math.Sqrt(Sum);
        }

        private double Distance(double winnerCoordinate, double Coordinate)
        {
            return Math.Sqrt(Sqr(winnerCoordinate - Coordinate));
        }

        private bool NaNCheck(NeuroNet net)
        {
            for (int i = 0; i < net.neurons.Count; i++)
            {
                for (int g = 0; g < net.neurons[i].weights.Count; g++)
                {
                    if (net.neurons[i].weights[g].Equals(System.Double.NaN))
                        return true;
                }
            }
            return false;
        }

        private double[] InitializeEuclideanDistanceSum(double[]
EuclideanDistanceSum, NeuroNet net, VectorsSet EdVectorSet)
```

```
        {
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                EuclideanDistanceSum[i] = 0;
            }
            return EuclideanDistanceSum;
        }

        private bool Changes(ref double[] OldEuclideanDistanceSum, NeuroNet
net, VectorsSet EdVectorSet, int n)
        {
            bool IsChanges = false;
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                EdVectorSet.vectors[i].BMUIndex = Test(net,
EdVectorSet.vectors[i].InputVector);
            }
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                if (Math.Sqrt(Sqr(OldEuclideanDistanceSum[i] -
EuclideanDistance(net.neurons[EdVectorSet.vectors[i].BMUIndex],
EdVectorSet.vectors[i].InputVector))) > 0.01)
                {
                    IsChanges = true;
                }
                OldEuclideanDistanceSum[i] =
EuclideanDistance(net.neurons[EdVectorSet.vectors[i].BMUIndex],
EdVectorSet.vectors[i].InputVector);
            }
            return IsChanges;
        }

        private bool Err(NeuroNet net, VectorsSet EdVectorSet, int n, ref
ProgressForm PF)
        {
            bool Error = true;
            double dist = 0;
            string ErrText = "";
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                EdVectorSet.vectors[i].BMUIndex = Test(net,
EdVectorSet.vectors[i].InputVector);
            }
            for (int i = 0; i < EdVectorSet.vectors.Count; i++)
            {
                    if (n == System.Int32.MaxValue)
                        return true;
                    dist =
EuclideanDistance(net.neurons[EdVectorSet.vectors[i].BMUIndex],
EdVectorSet.vectors[i].InputVector);
                    ErrText += "Разность " + i + "-гообучающеговектораи " +
EdVectorSet.vectors[i].BMUIndex + "-гонейрона" + ": " + dist + "\n";
                    if (dist > 0.001)
                    {
                        Error = false;
                        //break;
                    }
            }
            PF.ProgressClustersText = ErrText;
            return Error;
        }
```

```
        private double Sqr(double value)
{
            return value * value;
        }
    }
}
```

Приложение 6. Исходный код программы для ЭВМ. Матричный способ кластеризации беспроводной сенсорной сети

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace NeuroSens
{
    class TMatrix
    {
        int rows;
        int cols;
        int[, ,] matrix;

        public int Rows()
        {
            return rows;
        }
        public int Cols()
        {
            return cols;
        }

        public TMatrix(int rows = 0, int cols = 0)
        {
            this.rows = rows;
            this.cols = cols;
            this.matrix = new int [rows, cols, 3];
            for (int i = 0; i < rows; i++)
                for (int j = 0; j < cols; j++)
                {
                    this.matrix[i, j, 0] = 0;
                    this.matrix[i, j, 1] = i;
                    this.matrix[i, j, 2] = j;
                }
        }

        public int this[int rowIndex, int ColIndex, string type = "value"]
        {
            get
            {
                if (type.Equals("row"))
                    return this.matrix[rowIndex, ColIndex, 1];
                if (type.Equals("col"))
                    return this.matrix[rowIndex, ColIndex, 2];
                return this.matrix[rowIndex, ColIndex, 0];
            }
            set
            {
                this.matrix[rowIndex, ColIndex, 0] = value;
            }
        }
```

```
        public void AddIntersect()
        {
            if (rows > 0 || cols > 0)
            {
                int[, ,] TMPmatrix = new int[rows, cols, 3];
                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols; j++)
                        for (int k = 0; k < 3; k++)
                            TMPmatrix[i, j, k] = this.matrix[i, j, k];
                rows++;
                cols++;
                this.matrix = new int[rows, cols, 3];
                for (int i = 0; i < rows - 1; i++)
                    for (int j = 0; j < cols - 1; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i, j, k];
                for (int j = 0; j < cols; j++)
                    this.matrix[rows - 1, j, 1] = this.matrix[rows - 2, j, 1]
+ 1;
                for (int i = 0; i < rows; i++)
                    this.matrix[i, cols - 1, 2] = this.matrix[i, cols - 2, 2]
+ 1;
            }
            else
            {
                rows++;
                cols++;
                this.matrix = new int[rows, cols, 3];
            }
        }

        public void AddRow()
        {
            if (rows > 0 && cols > 0)
            {
                int[, ,] TMPmatrix = new int[rows, cols, 3];
                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols; j++)
                        for (int k = 0; k < 3; k++)
                            TMPmatrix[i, j, k] = this.matrix[i, j, k];
                rows++;
                this.matrix = new int[rows, cols, 3];
                for (int i = 0; i < rows - 1; i++)
                    for (int j = 0; j < cols - 1; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i, j, k];
                for (int j = 0; j < cols; j++)
                    this.matrix[rows - 1, j, 1] = this.matrix[rows - 2, j, 1]
+ 1;
                for (int i = 0; i < rows; i++)
                    this.matrix[i, cols - 1, 2] = this.matrix[i, cols - 2, 2]
+ 1;
            }
            else
            {
                rows++;
                this.matrix = new int[rows, cols, 3];
            }
        }

        public void AddColumn()
        {
```

```
            if (rows > 0 && cols > 0)
            {
                int[, ,] TMPmatrix = new int[rows, cols, 3];
                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols; j++)
                        for (int k = 0; k < 3; k++)
                            TMPmatrix[i, j, k] = this.matrix[i, j, k];
                cols++;
                this.matrix = new int[rows, cols, 3];
                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols - 1; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i, j, k];
                for (int j = 0; j < cols; j++)
                    this.matrix[rows - 1, j, 1] = this.matrix[rows - 2, j, 1]
+ 1;
            }
            else
            {
                cols++;
                this.matrix = new int[rows, cols, 3];
            }
        }

        public void DeleteIntersect(int row, int col)
        {
            int rowIndex = -1;
            int colIndex = -1;
            for (int i = 0; i < rows; i++)
                if (this.matrix[i, 0, 1] == row)
                {
                    rowIndex = i;
                    break;
                }

            for (int j = 0; j < cols; j++)
                if (this.matrix[0, j, 2] == col)
                {
                    colIndex = j;
                    break;
                }

            int[, ,] TMPmatrix = new int[rows, cols, 3];
            for (int i = 0; i < rows; i++)
                for (int j = 0; j < cols; j++)
                    for (int k = 0; k < 3; k++)
                        TMPmatrix[i, j, k] = this.matrix[i, j, k];

            if (rowIndex != -1 && colIndex != -1)
            {
                rows--;
                cols--;
                this.matrix = new int[rows, cols, 3];
                for (int i = 0; i < rowIndex; i++)
                    for (int j = 0; j < colIndex; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i, j, k];
                for (int i = rowIndex; i < rows; i++)
                    for (int j = colIndex; j < cols; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i + 1, j + 1,
k];
```

```
                for (int i = 0; i < rowIndex; i++)
                    for (int j = colIndex; j < cols; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i, j + 1, k];
                for (int i = rowIndex; i < rows; i++)
                    for (int j = 0; j < colIndex; j++)
                        for (int k = 0; k < 3; k++)
                            this.matrix[i, j, k] = TMPmatrix[i + 1, j, k];
            }
        }
    }

    class MatrixMethod
    {
        public List<int> GetClusterHeads2(List<List<int>> Clusters, int[,]
Matrix)
        {
            int N = Convert.ToInt32(Math.Sqrt(Matrix.Length));
            List<int> ClusterHeads = new List<int>();

            for (int i = 0; i < Clusters.Count; i++) // перебор кластеров
            {
                if (Clusters[i].Count > 0)
                {
                    double MaxAverage = 0;
                    int MaxAverageNode = Clusters[i][0];
                    for (int j = 0; j < Clusters[i].Count; j++) // перебор
узлов в кластере
                    {
                        double Average = 0;
                        for (int k = 0; k < N; k++) // перебор компонентов
узла
                        {
                            Average += Matrix[Clusters[i][j], k];
                        }
                        if (Average > MaxAverage)
                        {
                            MaxAverage = Average;
                            MaxAverageNode = Clusters[i][j];
                        }
                    }
                    ClusterHeads.Add(MaxAverageNode);
                }
            }

            return ClusterHeads;
        }

        public List<List<int>> GetClusters(int[,] Matrix1, bool Greedy, int
NodesPerCluster, ref List<int> ClusterHeads)
        {
            List<List<int>> Clusters = new List<List<int>>();
            int N = Convert.ToInt32(Math.Sqrt(Matrix1.Length));
            TMatrix Matrix = new TMatrix(N,N);

            ProgressForm PF = new ProgressForm();
            PF.Show();
            PF.ProgressPersent = 0;
            PF.ProgressText = "Подготовкаккластеризацииузлов...";
float Sum = 0;

            for (int i = 0; i < N; i++)
```

```
            {
                for (int j = 0; j < N; j++)
                {
                    Matrix[i, j] = Matrix1[i, j];
                }
            }

            int ClusteredItems = 0;

            while (ClusteredItems != N)
            {
                Analyse(ref Matrix, ref Clusters, ref ClusteredItems, N,
Greedy, NodesPerCluster);
                Sum = ((float)100 / (float)N * (float)ClusteredItems);
                PF.ProgressPersent = Convert.ToInt32(Math.Floor(Sum));
                PF.ProgressText = "Кластеризовано " + ClusteredItems + "
узлов из " + N;
            }

            ClusterHeads = GetClusterHeads2(Clusters, Matrix1);

            PF.ProgressPersent = 100;
            PF.ProgressText = "Кластеризация завершена";
PF.Hide();
PF.Dispose();
            return Clusters;
        }

        public void Analyse(ref TMatrix Matrix, ref List<List<int>> Clusters,
ref int ClusteredItems, int N, bool Greedy, int NodesPerCluster)
        {
            List<int> row = new List<int>();
            int MatrixSize = Matrix.Cols();
            int[] MaxSignalIndex = {0, 0}; // примем за элемент с максмальным
сигналом, первый элемент ММ
            intMaxSignal = -1; // примем за максимальный сигнал, единичный
отрицательный сигнал
for (inti = 0; i<MatrixSize; i++) // находим в цикле максимальный сигнал,
если есть одинаковые максимальные сигналы, то выбираем первый
{
                for (int j = 0; j < MatrixSize; j++)
                {
                    if ((Matrix[i, j] > MaxSignal) && Matrix[i, j] != 999)
                    {
                        MaxSignalIndex[0] = i; MaxSignalIndex[1] = j;
                        MaxSignal = Matrix[i, j];
}
                }
            }
if (MaxSignal == 0) // если максимальный сигнал равен 0 (обычно это в конце
кластеризации), то добавляем такой узел в отдельный отрезанный кластер
{
                row = new List<int>();
                row.Add(Matrix[MaxSignalIndex[0], MaxSignalIndex[1], "row"]);
                Clusters.Add(row);
                ClusteredItems++;
                DelIntersect(ref Matrix, ref Clusters);
                return;
            }
if (ClusteredItems + 1 == N) // ищем последний элемент и помещаем его в
отдельный кластер
{
```

```
                row = new List<int>();
                for (int i = 0; i < N; i++)
                    row.Add(i);
                for (int i = 0; i < Clusters.Count; i++)
                    for (int j = 0; j < Clusters[i].Count; j++)
                        row.Remove(Clusters[i][j]);

                Clusters.Add(row);
                ClusteredItems++;
                DelIntersect(ref Matrix, ref Clusters);
                return;
            }

            // помещаем строку с максимальным сигналом в отдельный массив,
который будем анализировать
TMatrix Sample = new TMatrix(2,0);

            //int[,] Sample = new int[2, MatrixSize];
            int SampleCount = 0;
            for (int j = 0; j < MatrixSize; j++)
            {
                if (Matrix[MaxSignalIndex[0], j] != 999)
                {
                    Sample.AddColumn();
                    Sample[0, SampleCount] = Matrix[MaxSignalIndex[0], j];
                    Sample[1, SampleCount] = Matrix[MaxSignalIndex[0], j,
"col"];
                    SampleCount++;
                }
            }

            // сортируем элементы в результирующей строке.
quickSort(ref Sample, 0, SampleCount - 1);

            int IntervalSum = 0;
            for (int i = 0; i < SampleCount - 1; i++)
            {
                IntervalSum += Sample[0, i] - Sample[0, i + 1];
            }
            int AverageSignal = Sample[0, 0];
            if (SampleCount != 1)
                AverageSignal = IntervalSum / (SampleCount - 1);

            row = new List<int>();
            row.Add(Matrix[MaxSignalIndex[0], MaxSignalIndex[1], "row"]);
            ClusteredItems++;
            row.Add(Matrix[MaxSignalIndex[0], MaxSignalIndex[1], "col"]);
            ClusteredItems++;
            if (SampleCount != 1)
            {
                if (Greedy)
                {
                    for (int i = 0; i < SampleCount - 1; i++)
                    {
                        if (NodesPerCluster > 0)
                            if (row.Count >= NodesPerCluster)
                                break;
                        if (Sample[0,i + 1] != 0)
                        {
                            if (Sample[1,i + 1] != Matrix[MaxSignalIndex[0],
MaxSignalIndex[1], "col"])
                            {
```

```
                                ClusteredItems++;
                                row.Add(Sample[1, i + 1]);
                            }
                        }
                        else
                            break;
                    }
                }
                else
                {
                    for (int i = 0; i < SampleCount - 1; i++)
                    {
                        if ((Sample[0,i] - Sample[0, i + 1]) < AverageSignal)
                        {
                            if (Sample[1, i + 1] != Matrix[MaxSignalIndex[0],
MaxSignalIndex[1], "col"])
                            {
                                ClusteredItems++;
                                row.Add(Sample[1, i + 1]);
                            }
                        }
                        else
                            break;
                    }
                }
            }
            Clusters.Add(row);

            DelIntersect(ref Matrix, ref Clusters);

        }

        public static void DelIntersect(ref TMatrix Matrix, ref
List<List<int>> Clusters)
        {
            int Last = Clusters.Count - 1;
            for (int k = 0; k < Clusters[Last].Count; k++)
            {
                Matrix.DeleteIntersect(Clusters[Last][k], Clusters[Last][k]);
            }
        }


        public void quickSort(ref TMatrix a, int l, int r)
        {
            int temp;
            int tempIndex;
            int x = a[0, l + (r - l) / 2];
//запись эквивалентна (l+r)/2,
            //но не вызввает переполнения на больших данных
int i = l;
            int j = r;
//код в while обычно выносят в процедуру particle
while (i <= j)
            {
                while (a[0, i] > x) i++;
                while (a[0, j] < x) j--;
                if (i <= j)
                {
                    temp = a[0, i];
                    tempIndex = a[1, i];
                    a[0, i] = a[0, j];
```

```
                    a[1, i] = a[1, j];
                    a[0, j] = temp;
                    a[1, j] = tempIndex;
                    i++;
                    j--;
                }
            }
            if (i < r)
                quickSort(ref a, i, r);

            if (l < j)
                quickSort(ref a, l, j);
        }

    }
}
```