Оглавление

Оглавление.....2
Введение..... 3
Основная часть..... 5
1 Неградиентный поиск экстремума по точечным значениям поля.....5
2 Классический (централизованный) метод роя частиц.....7
3 Сравнение с градиентным подъёмом..... 13
4 Децентрализованный метод роя частиц..... 15
5 Исследование робастности алгоритма..... 17
Список использованных источников..... 19
Приложение А.....20
Приложение Б.....36

Введение

В данной работе рассматривается задача управления мобильными роботами с целью выведения их в точку (или ее малую окрестность) физического пространства, в которой некоторое скалярное поле достигает своего экстремума, при этом поле априори неизвестно. Примерами таких полей могут служить: концентрация опасного химического агента в среде обитания, соленость морской воды, распределение температуры и т.д. Как правило, ввиду больших размеров зоны поиска применение распределенных по этой зоне ансамблей статических сенсоров требует высокой плотности распределения и большого числа сенсоров, что часто ставит под сомнение эффективность данного подхода. Использование мобильных роботов, напротив, является привлекательной стратегией: небольшое число роботов, способных проводить измерения, может исследовать достаточно большую область определения поля за разумное время.

Многие подходы к решению задачи поиска экстремума используют информацию о градиенте поля [1, 2]. Поскольку градиент редко доступен непосредственному измерению, большинство методов этой группы используют его оценки, построенные на основе одновременного измерения значения поля в нескольких точках пространства, образующих его аффинный базис. Такой способ адекватен при использовании крупноформатных роботов с большим числом датчиков или группы роботов, обменивающихся данными друг с другом [3, 4].

Тем не менее, если такой подход для нас неудобен, или если мы работаем над созданием альтернативных алгоритмов, имеет смысл обратить

внимание на ***биомиметику***. Биомиметика — это метод создания материалов и устройств, при котором ученые находят удачные идеи в природе и заимствуют их для своих разработок. Преимущество такого подхода в том, что решения, которыми пользуется животный и растительный мир, как правило, никем не запатентованы, поэтому они бесплатны для исследователей. К тому же за многолетнюю историю эволюции в природе закрепились определенные структуры и механизмы, которые помогают существам выживать в разных условиях. Поэтому ученые могут ориентироваться на самые эффективные решения природы, видоизменять их под задачи проектирования и создавать разных роботов, похожих на представителей окружающего мира.

Именно таким образом был разработан ***метод роя частиц*** (Particle Swarm Optimization, PSO) [5]. Он основан на моделировании поведения стаи птиц или роя насекомых. Идея заключается в том, что каждый робот в группе рассматривается как "частица", которая движется в пространстве состояний в поисках оптимального решения. Частицы обмениваются информацией о своих текущих положениях и наилучших найденных решениях, что позволяет им совместно искать оптимальное решение для всей группы. Преимущество метода роя частиц заключаются в его способности к адаптации и поиску оптимального решения в сложных и динамичных средах. Благодаря возможности обмена информацией и координации между роботами, группа может эффективно справляться с задачами, такими как поиск, исследование или патрулирование. В данной статье мы рассмотрим применение нескольких версий метода роя частиц для управления группой мобильных роботов и его потенциальные преимущества. Будут представлены результаты экспериментов и анализ эффективности данного подхода.

Основная часть

1 Неградиентный поиск экстремума по точечным значениям поля

Пусть на плоскости определено скалярное поле D(x, y) ∈ R. Задача заключается в поиске его экстремума, где под поиском понимается физическое перемещение автономного робота или группы роботов в точку экстремума. При этом ни один из них не способен измерять градиент поля, однако каждый робот измеряет значение поля в своей текущей локации. Помимо этого, роботы способны к коммуникации друг с другом в пределах некоторого радиуса связи, либо роботы способны коммуницировать с некоторым центром. Конкретный тип связи, используемый роботами, будет уточнён ниже.

Для всех наших экспериментов мы использовали поле размера 10x10 условных единиц, определяемое формулой:

$$D(x, y) = \frac{1}{10\sqrt{2\pi}} e^{-\left(\frac{(x-5)^2}{10} + \frac{(y-5)^2}{10}\right)} \quad (1.1)$$

На изображениях ниже вы можете наблюдать линии уровня данной функции и поверхность, которую она описывает:

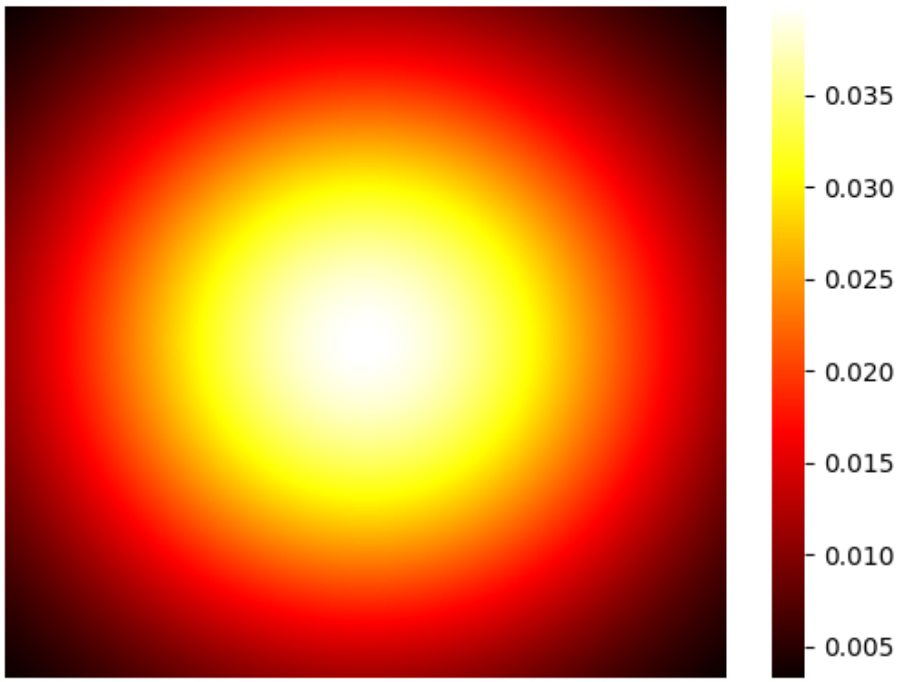

Рисунок 1.1 – линии уровня функции (1.1)

Как вы можете видеть, наша целевая функция представляет собой двумерную функцию Гаусса, чей центр был смещён в точку (5, 5), а дисперсия равняется 10, что обеспечивает достаточно медленное убывание функции.

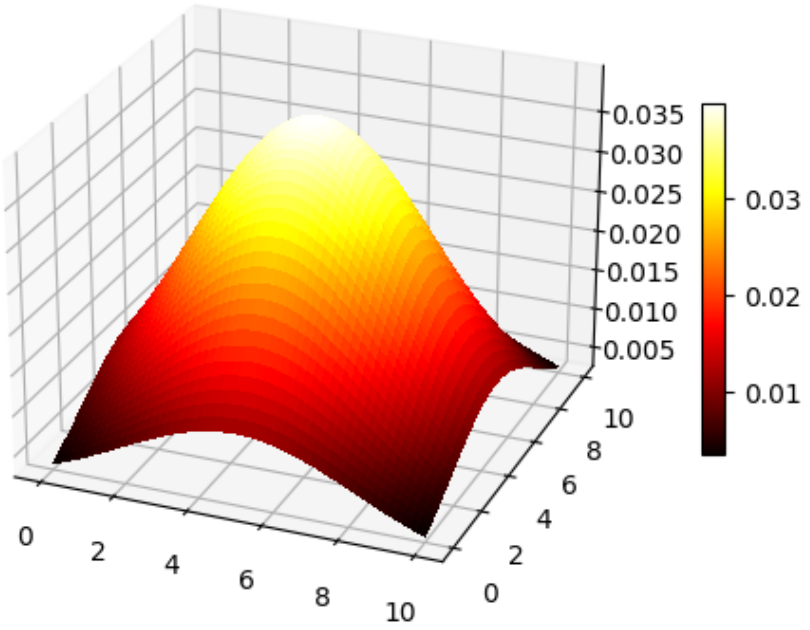

Рисунок 1.2 – поверхность, описываемая функцией (1.1)

В силу тех ограничений, что мы наложили на функционал роботов выше, мы не может измерять значение градиента. Тем не менее, вполне традиционный выход из такой ситуации — построение оценок градиента. Это, однако, требует доступа к значениям поля в нескольких близких точках, распределенных в двух измерениях. Группа из трех и более роботов может получить доступ к таким значениям (при условии адекватного взаимного позиционирования роботов). Этот факт лежит в основе неградиентных алгоритмов поиска экстремума группами общающихся друг с другом роботов.

Однако, данная работа нацелена на разработку и анализ альтернативных методов поиска экстремума, мотивированных биологическими прототипами.

2 Классический (централизованный) метод роя частиц

Метод роя частиц был предложен группой американских исследователей в 1995 года [5]. Через 3 года те же авторы опубликовали его улучшенную версию [6]. За последующие 25 лет было предложено несколько модификаций, мы же будет ориентироваться на версию 1998 года.

Псевдокод этого алгоритма представлен ниже:

1. Для каждого агента i = 1, ..., S выполнить
    a. Проинициализировать положение агента вектором из многомерного равномерного распределения: $x_i \sim \mathrm{U}(b_{\text{нижн.}}, b_{\text{верхн.}})$
    b. Проинициализировать лучшее положение, известное агенту, его текущей позицией: $p_i \leftarrow x_i$
    c. Если $f(p_i) > f(g)$, тогда обновить лучшее положение роя: $g \leftarrow p_i$
    d. Проинициализировать начальную скорость агента:
    $$v_i \sim \mathrm{U}(-|b_{\text{верхн.}} - b_{\text{нижн.}}|, |b_{\text{верхн.}} - b_{\text{нижн.}}|)$$
2. Пока не выполнен критерий останова выполнить:
    a. Для каждого агента i = 1, ..., S выполнить:
        i. Выбрать случайные величины $r_p, r_g \sim \mathrm{U}(0, 1)$
        ii. Обновить скорость агента:
        $$v_i \leftarrow w v_i + c_1 r_p (p_i - x_i) + c_2 r_g (g - x_i)$$
        iii. Обновить положение агента:
        $$x_i \leftarrow x_i + v_i$$
        iv. Если $f(p_i) > f(x_i)$, обновить лучшее положение, известное агенту: $p_i \leftarrow x_i$
        v. Если $f(p_i) > f(g)$, обновить лучшее положение, известное рою: $g \leftarrow p_i$

Использованные выше обозначения:

- S – количество агентов;
- p_i – лучшее положение, известное агенту i;
- g – лучшее положение, известное рою;
- w – параметр инерции, обычно 1;
- c_1 и c_2 – множители перед теми компонентами скорости, что генерируются лучшей позицией частицы и лучшей позицией роя, обычно 2;
- $b_{\text{нижн.}}$ и $b_{\text{верхн.}}$ – ограничения снизу и сверху соответственно;

Поскольку этот метод был предложен именно как алгоритм оптимизации, к управлению группой роботов он пока не совсем применим:

1. Агенты могут иметь начальное положение в центре поля, что невозможно реализовать физически, ведь робот не может просто материализовать посреди объекта;
2. Скорость агента вполне может быть такой, что он за одну итерацию преодолеет всё поле, что может иметь место только в случае поля небольшого размера (в пределах нескольких десятков метров);
3. Как правило, количество агентов довольно большое. Считается, что оптимально их количество – это 20 – 50 [7]. Едва ли такое количество роботов можно себе позволить на практике.
4. Выбор параметра w, который отвечает за инерцию, очень важен. Если взять его слишком большим, то алгоритм будет расходиться.
5. Обычно критерий останова в этом алгоритме - это количество итераций, что может быть слишком грубым условием;
6. В данном алгоритме агенты могут коммуницировать с некоторой базой – сервером, который хранит лучшее найденное роем положение. Реализовать такой механизм может быть затратно, а иногда и невозможно;

Поэтому нами был предложен ряд модификаций:

1. В начальный момент времени роботы расположены по периметру поля:
    a. Агенты располагаются независимо друг от друга в произвольном месте на периметре поля. Эту стратегию далее мы именуем «***edge***»;
    b. Все агенты располагаются в небольшом “пятне”, расположенном на периметре поля. Эту стратегию далее мы именуем «***spot***»;
2. Если модуль скорости превышает размер поля, уменьшенный в 50 раз. то мы производим нормировку скорости так, чтобы её модуль стал равен размеру поля, уменьшенному в 50 раз. Таким образом, в лучшем случае агент преодолеет всё поле за 50 итераций. Такое ограничение выбрано из тех соображений, что гражданский квадрокоптер средней ценовой категории, например, «DJI Mavic 3», может развивать скорость до 20 м/с. В то же время, более серьёзные БПЛА, например, «Орлан-30», могут развивать скорость до 30 м/с. Таким образом, мы можем моделировать работу алгоритма на поле, чей линейный размер составляет несколько километров;
3. Здесь мы нацелены на реальное применение алгоритма, поэтому будем использовать небольшое количество агентов: от 2 до 20;
4. Каждые 100 итераций мы уменьшаем w на 25%: $w \leftarrow w * 0.75$. Это достаточно сильно улучшает сходимость;
5. Помимо количества итераций, алгоритм останавливается, если норма скорости стала слишком маленькой, либо, если 75% агентов находятся в малой окрестности некоторой точки;
6. Мы также рассмотрим версии алгоритма, в которых агенты общаются напрямую друг с другом, без сервера, в пределах некоторого радиуса связи. Также было бы интересно изучить влияние радиуса связи на результаты работы алгоритма.

Для большей наглядности, ниже продемонстрированы положения роя в начальный момент времени при различных стратегиях:

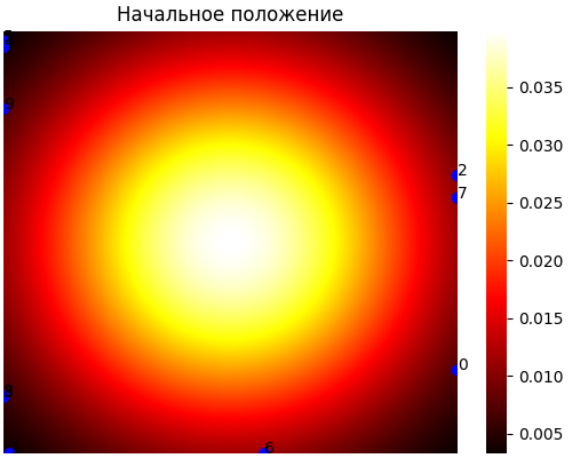

Рисунок 2.1 – Стратегия «edge»

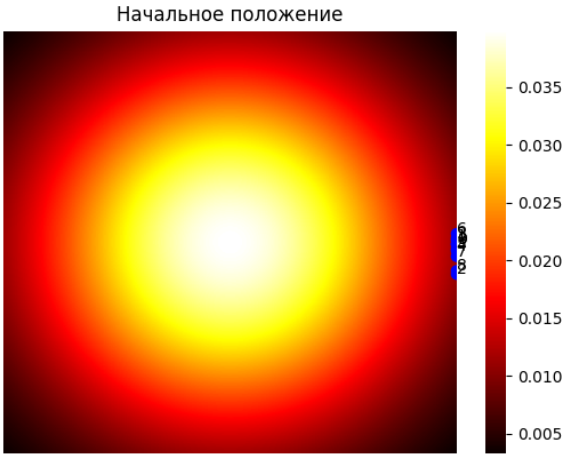

Рисунок 2.2 – Стратегия «spot»

Для сравнения различных версий метода роя частиц мы будем использовать метод Монте-Карло [8]. А именно, мы будем запускать алгоритм 1000 раз для каждой из его вариаций и логировать такие метрики, как: количество итераций – аналог времени работы, абсолютную и относительную ошибку алгоритма, путь, пройденный всем роем, и средний путь, пройденный агентом – последние две метрики показывают, какое количество абстрактного топлива требуется для работы алгоритма.

Для начала исследуем централизованный алгоритм – то есть такой, где агенты общаются через сервер, а не напрямую. Соответственно, к методу, чей псевдокод представлен выше, применим наши модификации 1. – 5., без пункта 6. – о децентрализованном алгоритме речь пойдёт ниже. На графиках ниже точкой отмечено среднее значение метрики по итогам симуляции, а вертикальные линии – это стандартное отклонение. Синий цвет соответствует стратегии начального расположения агентов «edge», оранжевый – стратегии «spot». Мы будем варьировать количество агентов, которые участвуют в поиске максимума – в идеале, мы хотим использовать как можно меньше роботов, сохранив, при этом, хороший результат.

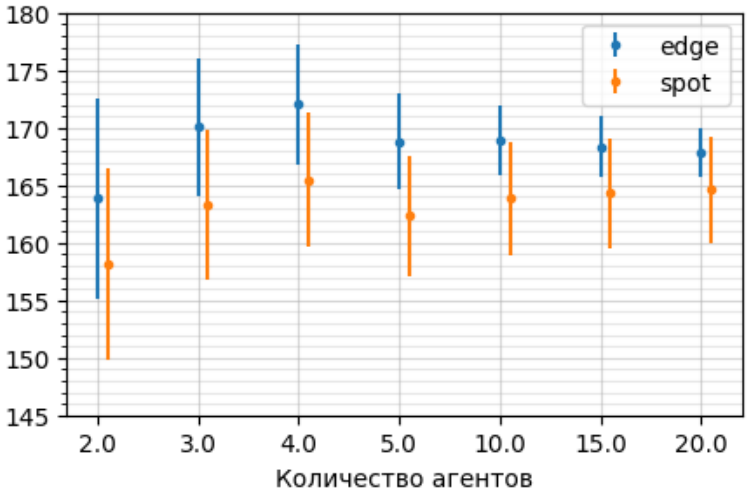

Рисунок 2.3 – количество итераций для различных версий централизованного алгоритма

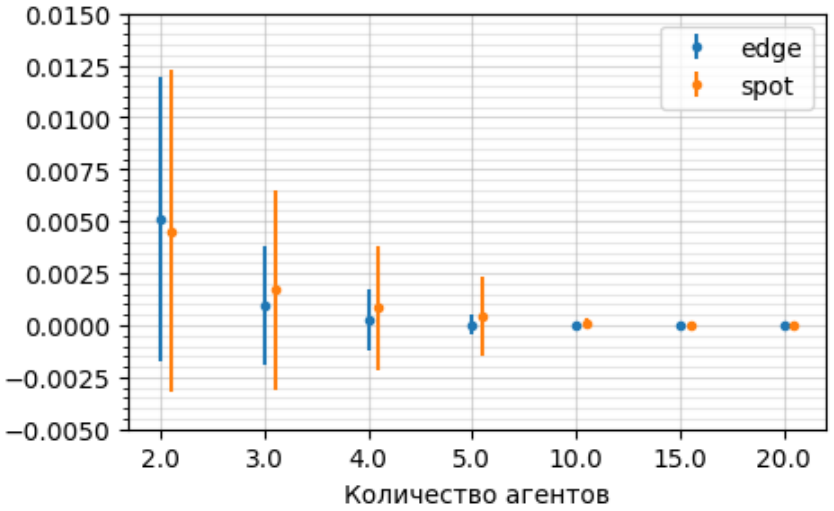

Рисунок 2.4 – абсолютная ошибка для различных версий централизованного алгоритма

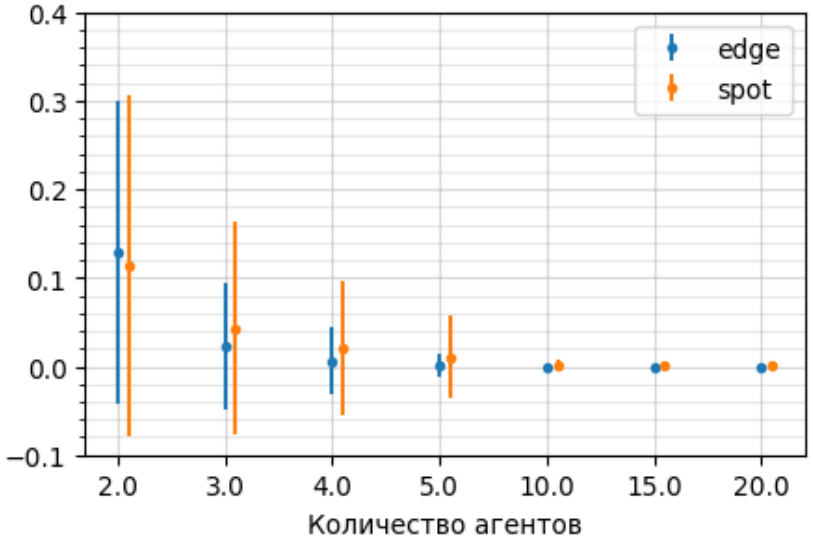

Рисунок 2.5 – относительная ошибка (в долях) для различных версий централизованного алгоритма

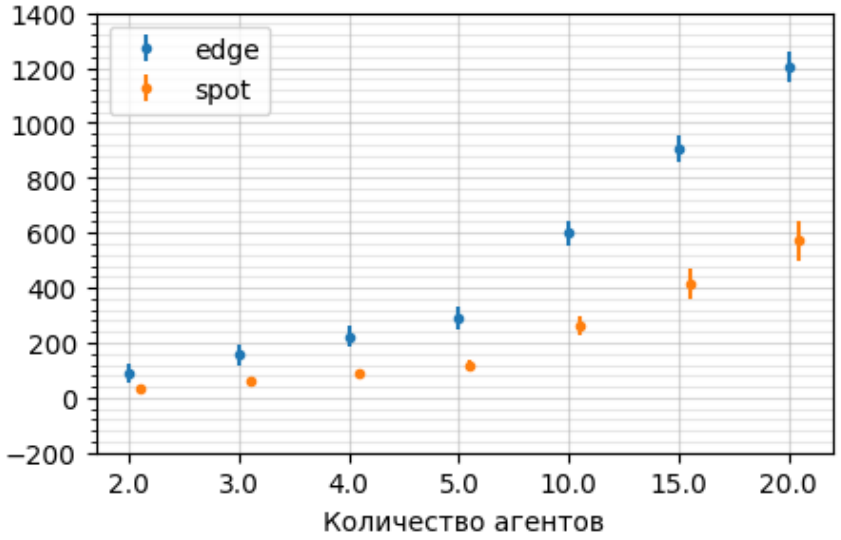

Рисунок 2.6 – суммарный путь, пройденный роем, для различных версий централизованного алгоритма

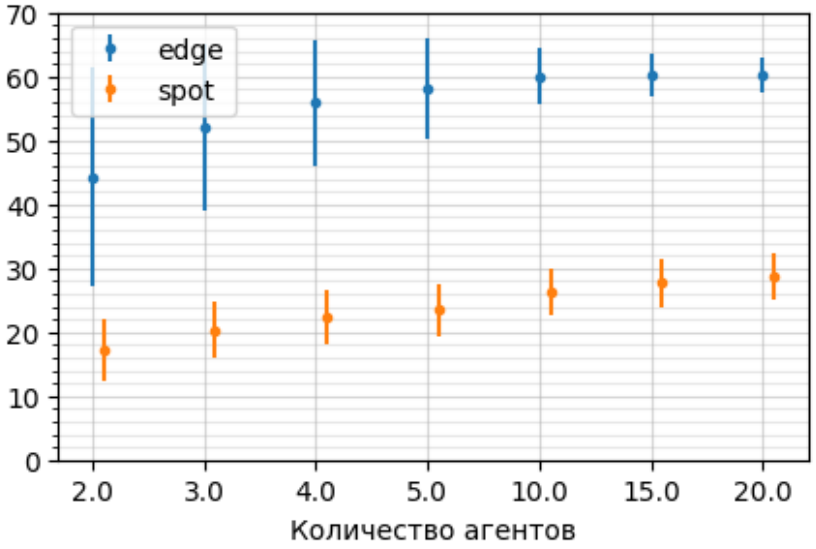

Рисунок 2.7 – средний путь, пройденный агентом, для различных версий централизованного алгоритма

Из рисунков 2.3 – 2.7 мы можем сделать следующие выводы:

1. Начиная с 10 агентов, алгоритм работает безошибочно при любой начальной стратегии. Если же в нашем распоряжении только 5 роботов, имеет смысл использовать стратегию «edge» – тогда ошибка составит не более 2%. Использование меньшего количества агентов нецелесообразно – при такой конфигурации алгоритм довольно сильно ошибается.
2. Стратегия «spot» показывает более слабый результат, чем стратегия «edge». Это связано с тем, что, находясь на, скажем, противоположных

краях карты, при взаимодействии агентов происходит следующее: на предыдущей итерации 1-ый робот находит точку, которую рой признаёт наилучшей – на текущей итерации 2-ой робот имеет компоненту скорости, направленную к этой точке, эта компонента скорости заставляет 2-ого агента двигаться по направлению к центру, поскольку роботы расположены на противоположных сторонах поля. Из-за вида целевой функции – она имеет максимум в центре поля, 2-ой агент находит точку, более близкую к центру поля, чем та, из которой он начинал текущую итерацию. Однако, этот эффект нивелируется с увеличением числа агентов – из-за случайной инициализации скорости при большом количестве агентов они всё же способны грамотно исследовать поле и в дальнейшем найти точку максимума целевой функции.

3. На лицо ожидаемый trade-off: за большую точность приходится платить большим расходом топлива – средний путь, пройденный агентов, растёт с увеличением числа самих агентов.
4. Если у нас в распоряжении есть хотя бы 10 агентов, имеет смысл использовать стратегию «spot», ведь она, как и стратегия «edge», гарантирует нам нахождение искомой точки, в то время, как количество итераций будет на 5-10 меньше, а пройденный путь уменьшится более, чем в 2 раза.

3 Сравнение с градиентным подъёмом

Как мы выяснили выше, наш алгоритм хорошо показал себя, если в рое есть хотя бы 5 агентов. Однако, поскольку он разрабатывается, как конкурент градиентным методам, хотелось бы сравнить качество работы с одним из их представителей, например, градиентным подъёмом. Для этого воспользуемся той же стратегией – методом Монте-Карло.

Таблица 2.1 – Сравнение градиентного подъёма и метода роя частиц со стратегией «edge»

Метрика	Алгоритм				
	Градиентный подъём	Централизованный метод роя частиц			
		2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	3.13e-07 ± 7.23e-10	1.286e-01 ± 1.717e-01	2.325e-02 ± 7.162e-02	6.232e-03 ± 3.766e-02	9.023e-04 ± 1.255e-02
Пройденный путь	5.76 ± 0.63	88.47090376 ± 34.23426822	155.851 ± 39.029	223.351 ± 39.686	290.768 ± 39.861
Количество итераций	2296.584 ± 159.23	163.849 ± 8.67	170.057 ± 5.945	172.052 ± 5.245	168.811 ± 4.136

Метрика	Алгоритм		
	Централизованный метод роя частиц		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	4.043e-07 ± 2.808e-06	1.638e-08 ± 1.339e-07	2.09e-09 ± 1.272e-08
Пройденный путь	599.589 ± 44.367	904.923 ± 49.657	1202.836 ± 55.715
Количество итераций	168.936 ± 2.994	168.361 ± 2.609	167.773 ± 2.125

Выводы, которые мы можем сделать из этой таблицы:

1. Начиная с 10 агентов, алгоритм демонстрирует сравнимую с градиентным подъёмом точность, а при увеличении числа роботов даже превосходит по этому параметру своего конкурента;
2. Снова возникает старый trade-off: при сопоставимой точности наш алгоритм требует в 87 – 126 больше топлива на передвижение. В то же время, он требует в 12 – 14 раз меньше итераций, а поскольку в реальности агенты двигаются независимо друг от друга и одновременно, выигрыш по времени также составит 12 – 14 раз;

4 Децентрализованный метод роя частиц

Как я уже писал выше, далеко не всегда мы хотим или даже можем использовать некий сервер, который можем общаться с каждым из агентов и хранить некую информацию, которая от них поступает. Именно по этой причине хотелось бы промоделировать поведение группы роботов в ситуации, когда агенты могут связываться друг с другом напрямую, но только в пределах некоторого ограниченного радиуса связи. Алгоритм, по которому в таком случае группа ищет максимум, мы назвали **«Децентрализованный метод роя частиц»**. Его главные отличия от модифицированного алгоритма, что мы описали в разделе 2 текущей главы:

1. Вместо общего сервера, который хранит наибольшее найденное значение и позицию, которая ему соответствует, теперь каждый робот хранит эту информацию внутри себя. То есть, у каждого агента теперь есть отдельная переменная, отвечающая за лучшее положение, найденное роем;
2. После этапа движения агент связывается со всеми другими агентами, что попадают в его радиус связи (включая самого этого агента). Если существуют агенты, что смогли найти точку, значение в которой

больше, чем то, которое агент смог получить от роя ранее, то агент обновляет свою собственную переменную, отвечающую за информацию. полученную от роя;

Ознакомиться с таблицами метрик, которые может достигнуть децентрализованный алгоритм в его различных конфигурациях, вы можете в Приложении А. Здесь же мы проанализируем эти данные и сделаем выводы о применимости алгоритма на практике:

1. Если вы используйте стратегию «edge», то минимальный радиус связи, при котором алгоритм способен гарантировать нахождение точки экстремума – это 5% от размера поля. В случае, если вы применяете стратегию «spot» (которая более реализуема на практике), то для аналогичного результата понадобится уже 10+ агентов;
2. Суммарный пройденный группой путь в среднем уменьшается при увеличении радиуса связи. Это может быть важно, если топливо в дефиците;
3. Количество итераций, которое можно считать аналогом времени, при увеличении радиуса связи незначительно возрастает для стратегии «edge» и заметно уменьшается для стратегии «spot». Если время работы довольно критично, то стоит обратить внимание на этот факт;
4. При уменьшении радиуса связи с 5% от размера поля алгоритм начинает терять в точности работы. Таким образом, даже при наибольшем числе агентов алгоритм ошибается в среднем на 2% и вплоть до 6% независимо от начального положения группы;

5 Исследование робастности алгоритма

Поскольку мы исследуем, насколько реально применим метод роя частиц для управлением группой роботов, стоит узнать, насколько агент может преодолеть шум, который появляется при измерении чего-либо датчиком. Для этого, при симуляции добавим шум в измерения при каждом обращении агента к полю.

Этот шум должен обладать следующими свойствами:

1. Чем дальше от экстремума находится агент, тем сильнее шум должен влиять на результаты измерений;
2. При наибольшем удалении от экстремума шум может достигать такой величины, что его значение превысит значение поля в искомой точке максимума. Таким образом, особо невезучий агент может остаться где-то на окраине поля до окончания работ. Более того, при достаточном радиусе связи он привлечёт в этот ложный сектор и других агентов, чем саботирует работу группы;

Ориентируясь на эти 2 условия, мы выбрали 2 варианта шума:

1. Шум из равномерного распределения
   $U(-1, 1)x||\text{положение агента} - (5, 5)||_2 \, x \, 0.0057$
2. Шум из нормального распределения
   $N(0, ||\text{положение агента} - (5, 5)||_2 \, x \, 0.0057)$

Понимая, что такой сильный шум способен серьёзно повлиять на работу алгоритма, мы также рассмотрим его ослабленные, чтобы понять, в какой момент алгоритм может прийти в негодность.

Таким образом, у нас получились следующие варианты шума:

1. Для равномерного распределения:
   a. $U(-1, 1)x||\text{положение агента} - (5, 5)||_2 \, x \, 0.001425$;
   b. $U(-1, 1)x||\text{положение агента} - (5, 5)||_2 \, x \, 0.00285$;
   c. $U(-1, 1)x||\text{положение агента} - (5, 5)||_2 \, x \, 0.004275$;

d. $U(-1, 1)x||\text{положение агента} - (5, 5)||_2 x\, 0.0057$

2. Для нормального распределения:
   a. $N(0, ||\text{положение агента} - (5, 5)||_2 x\, 0.00063)$
   b. $N(0, ||\text{положение агента} - (5, 5)||_2 x\, 0.00126)$
   c. $N(0, ||\text{положение агента} - (5, 5)||_2 x\, 0.00189)$

Далее мы исследуем, насколько наш децентрализованный алгоритм способен этому шуму противостоять. Возможно, агент – лидер окажется способным вывести группу из зашумлённого региона. Также возможно, что влияние шума скажется на поведении алгоритма самым плохим образом.

1. При слабом влиянии шума (1.a. 2.a) преодолевается нашим алгоритмом при 10+ агентах и радиусе связи хотя бы 100% от размера поля. Если же уменьшить радиус связи до 50%, то потребуется уже минимум 15 агентов. Если уменьшить его до 25%, то для корректной работы алгоритмы нужно будет по крайней мере 20 агентов. При дальнейшем уменьшении радиуса связи алгоритм быстро теряет в качестве работы;
2. При среднем влиянии шума (1.b-c, 2.b) алгоритм справляется с шумом при 10+ агентах с радиусов связи 100% от размера поля, но уже при радиусе связи равном 50% относительная ошибка вырастает в среднем до 4% и вплоть до 12 % при 20 агентах. При дальнейшем уменьшении радиуса связи алгоритм становится всё менее пригодным для решения задачи;
3. При сильном влиянии шума (1.d, 2.c) алгоритм ошибается в среднем на 8% и вплоть до 16% даже при 20 агентах и связью, покрывающей всё поля. При уменьшении числа агентов или уменьшении радиуса связи алгоритм ошибается всё сильнее.
4. Во всех случаях успешной работы алгоритма была использованы стартегия «edge»;

Список использованных источников

1. N. Atanasov, J. Le Ny, N. Michael, G.J. Pappas // Stochastic source seeking in complex environments: Proceedings of the IEEE Conference on Robotics and Automation / Saint Paul, MN, 2012, P. 3013–3018.

2. Zhang C., Ord´onez R. Extremum-seeking control and applications: A numerical optimization-based approach // Advances in Industrial Control. — Springer-Verlag, 2012.

3. Ogren P., Fiorelli E., Leonard N. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment // IEEE Trans. Autom. Control, vol. 49, no. 8, 2004, P. 1292–1301.

4. Peterson C., Paley D. Multivehicle coordination in an estimated time-varying flowfield // Journal of Guidance, Control, and Dynamics, vol. 34, no. 1, 2011. P. 177–191.

5. Kennedy, J.; Eberhart, R. // Particle Swarm Optimization: Proceedings of IEEE International Conference on Neural Networks. IV. P. 1942—1948.

6. Y. Shi, R. Eberhart. A modified particle swarm optimizer: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence / Anchorage, AK, USA, 1998, P. 69-73,

7. A.P. Piotrowski, J.J. Napiorkowski, Piotrowska A. E. Population size in Particle Swarm Optimization // Swarm and Evolutionary Computation, vol. 58, 2020,

8. N. Metropolis, S. Ulam. The Monte Carlo Method // Journal of the American Statistical Association, vol. 44, no. 247, 1949, P. 335–341

Приложение А

(обязательное)

Метрики различных вариаций децентрализованного алгоритма

Ниже будут представлены метрики децентрализованного алгоритма при различных значениях радиуса связи.

Таблица А.1 – метрики алгоритма при радиусе связи, равном 142% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.275e-01 ± 1.806e-01	1.941e-02 ± 6.648e-02	2.51e-03 ± 1.652e-02	6.647e-04 ± 5.661e-03
Пройденный путь	80.484 ± 28.806	147.477 ± 33.065	211.809 ± 37.897	275.239 ± 37.992
Количество итераций	163.576 ± 8.54	170.205 ± 6.15	172.035 ± 5.12	169.345 ± 4.25

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	8.443e-07 ± 8.458e-06	3.427e-08 ± 3.704e-07	2.15e-09 ± 1.307e-08
Пройденный путь	577.012 ± 41.772	874.062 ± 48.471	1171.336 ± 53.613
Количество итераций	168.908 ± 3.18	168.369 ± 2.76	167.873 ± 2.17

Таблица А.2 – метрики алгоритма при радиусе связи, равном 142% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.274e-01 ± 1.806e-01	1.941e-02 ± 6.648e-02	2.51e-03 ± 1.652e-02	6.646e-04 ± 5.661e-03
Пройденный путь	32.76 ± 8.995	57.349 ± 12.665	82.177 ± 16.034	107.958 ± 19.319
Количество итераций	158.225 ± 9.876	163.113 ± 6.314	164.95 ± 5.881	161.87 ± 5.427

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	8.443e-07 ± 8.458e-06	3.427e-08 ± 3.704e-07	2.147e-09 ± 1.307e-08
Пройденный путь	240.691 ± 36.997	375.914 ± 53.987	507.17 ± 71.856
Количество итераций	162.973 ± 4.614	162.891 ± 4.756	162.623 ± 4.72

Таблица А.3 – метрики алгоритма при радиусе связи, равном 100% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.279e-01 ± 1.744e-01	2.696e-02 ± 2.853e-02	4.113e-03 ± 2.485e-02	1.295e-03 ± 1.571e-02
Пройденный путь	86.857 ± 34.453	156.765 ± 38.692	225.463 ± 38.414	289.109 ± 39.235
Количество итераций	163.849 ± 8.67	170.057 ± 5.945	172.022 ± 5.119	168.992 ± 4.256

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	3.441e-07 ± 1.858e-06	1.69e-08 ± 1.927e-07	3.586e-09 ± 5.156e-08
Пройденный путь	600.151 ± 43.016	904.495 ± 49.894	1206.238 ± 55.642
Количество итераций	168.909 ± 3.169	168.302 ± 2.579	167.92 ± 2.209

Таблица А.4 – метрики алгоритма при радиусе связи, равном 100% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.14e-01 ± 1.912e-01	3.806e-02 ± 1.001e-01	1.551e-02 ± 5.256e-02	8.299e-03 ± 3.711e-02
Пройденный путь	34.486 ± 9.511	61.065 ± 13.2	88.063 ± 16.747	117.691 ± 20.954
Количество итераций	157.294 ± 8.589	163.383 ± 6.383	165.229 ± 5.705	162.49 ± 5.127

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	1.516e-03 ± 5.445e-03	7.201e-04 ± 3.767e-03	3.963e-04 ± 1.529e-03
Пройденный путь	263.314 ± 37.88	415.587 ± 52.949	574.301 ± 76.216
Количество итераций	164.051 ± 4.787	164.152 ± 4.67	164.327 ± 4.515

Таблица А.5 – метрики алгоритма при радиусе связи, равном 50% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.316e-01 ± 1.776e-01	2.463e-02 ± 7.429e-02	5.22e-03 ± 3.428e-02	5.445e-04 ± 5.964e-03
Пройденный путь	87.203 ± 33.788	156.595 ± 37.95	223.985 ± 39.13	289.269 ± 39.703
Количество итераций	163.947 ± 8.465	170.208 ± 6.017	172.092 ± 5.085	168.969 ± 4.271

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	2.822e-07 ± 1.719e-06	1.956e-08 ± 1.956e-07	3.383e-09 ± 5.08e-08
Пройденный путь	597.486 ± 46.008	905.04 ± 48.943	1204.276 ± 55.268
Количество итераций	168.727 ± 3.059	168.448 ± 2.475	167.924 ± 2.221

Таблица А.6 – метрики алгоритма при радиусе связи, равном 50% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.087e-01 ± 1.9e-01	3.873e-02 ± 1.103e-01	2.034e-02 ± 8.092e-02	9.331e-03 ± 4.683e-02
Пройденный путь	34.294 ± 9.302	60.74 ± 13.493	87.923 ± 16.262	117.107 ± 20.313
Количество итераций	157.587 ± 8.186	163.039 ± 5.98	165.228 ± 5.81	162.493 ± 5.254

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	1.68e-03 ± 7.027e-03	8.046e-04 ± 3.405e-03	4.374e-04 ± 2.294e-03
Пройденный путь	264.615 ± 37.965	416.426 ± 57.692	572.223 ± 74.77
Количество итераций	163.987 ± 4.767	164.162 ± 4.704	164.023 ± 4.488

Таблица А.7 – метрики алгоритма при радиусе связи, равном 25% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.369e-01 ± 1.881e-01	2.622e-02 ± 7.33e-02	5.473e-03 ± 3.825e-02	6.21e-04 ± 4.41e-03
Пройденный путь	87.141 ± 34.317	156.14 ± 38.118	222.643 ± 38.769	288.284 ± 41.264
Количество итераций	163.94 ± 8.63	170.263 ± 5.81	171.953 ± 4.98	168.708 ± 4.26

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	3.353e-06 ± 8.575e-05	5.173e-08 ± 9.1e-07	4.088e-09 ± 3.623e-08
Пройденный путь	601.807 ± 42.61	903.447 ± 48.837	1201.956 ± 53.669
Количество итераций	168.866 ± 3.11	168.291 ± 2.51	167.879 ± 2.14

Таблица А.8 – метрики алгоритма при радиусе связи, равном 25% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.224e-01 ± 2.01e-01	4.013e-02 ± 1.168e-01	1.613e-02 ± 6.582e-02	1.032e-02 ± 4.554e-02
Пройденный путь	34.247 ± 9.099	60.38 ± 12.996	88.292 ± 16.962	116.105 ± 19.439
Количество итераций	157.537 ± 8.17	163.214 ± 6.29	164.944 ± 5.66	162.378 ± 5.13

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	1.287e-03 ± 5.187e-03	7.502e-04 ± 3.556e-03	4.846e-04 ± 2.198e-03
Пройденный путь	263.947 ± 38.135	416.644 ± 53.546	572.627 ± 72.742
Количество итераций	163.999 ± 4.77	164.233 ± 4.63	164.225 ± 4.63

Таблица А.9 – метрики алгоритма при радиусе связи, равном 10% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.28e-01 ± 2	2.626e-02 ± 7.743e-02	5.785e-03 ± 3.496e-02	1.047e-03 ± 8.16e-03
Пройденный путь	87.925 ± 34.034	156.121 ± 39.637	221.628 ± 42.607	288.495 ± 38.325
Количество итераций	164.293 ± 8.459	169.945 ± 5.992	172.022 ± 4.859	168.941 ± 4.161

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	8.588e-07 ± 1.535e-05	1.077e-08 ± 8.844e-08	3.676e-09 ± 3.03e-08
Пройденный путь	600.068 ± 44.946	903.137 ± 48.801	1206.752 ± 56.41
Количество итераций	168.79 ± 3.058	168.217 ± 2.55	167.903 ± 2.22

Таблица А.10 – метрики алгоритма при радиусе связи, равном 10% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.198e-01 ± 2.001e-01	4.159e-02 ± 1.186e-01	1.963e-02 ± 7.432e-02	8.51e-03 ± 3.395e-02
Пройденный путь	34.732 ± 9.143	60.739 ± 13.076	88.617 ± 16.863	117.994 ± 20.706
Количество итераций	157.925 ± 8.397	163.1 ± 6.25	165.344 ± 5.721	162.626 ± 5.382

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	1.924e-03 ± 8.118e-03	6.846e-04 ± 2.968e-03	5.25e-04 ± 2.438e-03
Пройденный путь	265.791 ± 39.961	413.226 ± 56.356	573.278 ± 74.507
Количество итераций	163.886 ± 4.881	164.036 ± 4.716	164.354 ± 4.693

Таблица А.11 – метрики алгоритма при радиусе связи, равном 5% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.431e-01 ± 1.876e-01	2.783e-02 ± 8.965e-02	5.255e-03 ± 3.504e-02	4.642e-04 ± 3.511e-03
Пройденный путь	86.187 ± 32.966	154.797 ± 38.531	223.347 ± 39.475	289.934 ± 39.736
Количество итераций	163.397 ± 8.572	170.303 ± 5.813	172.036 ± 4.906	168.744 ± 4.064

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	4.983e-07 ± 5.74e-06	9.079e-09 ± 6.473e-08	3.03e-09 ± 3.388e-08
Пройденный путь	599.4 ± 42.491	900.085 ± 49.852	1204.595 ± 53.599
Количество итераций	169.009 ± 3.218	168.212 ± 2.525	168.001 ± 2.168

Таблица А.12 – метрики алгоритма при радиусе связи, равном 5% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	1.207e-01 ± 2.019e-01	4.595e-02 ± 1.237e-01	2.054e-02 ± 7.178e-02	8.783e-03 ± 4.051e-02
Пройденный путь	34.488 ± 9.254	61.171 ± 13.582	89.185 ± 16.543	116.681 ± 19.992
Количество итераций	157.489 ± 8.398	163.209 ± 6.567	165.292 ± 5.805	162.542 ± 5.012

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	1.919e-03 ± 1.156e-02	6.939e-04 ± 2.941e-03	5.277e-04 ± 2.312-03
Пройденный путь	264.594 ± 39.025	416.994 ± 55.059	570.117 ± 73.526
Количество итераций	164.087 ± 4.878	164.375 ± 4.778	164.468 ± 4.665

Таблица А.13 – метрики алгоритма при радиусе связи, равном 1% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	3.124e-01 ± 2.469e-01	2.301e-01 ± 2.206e-02	1.693e-01 ± 1.798e-01	1.337e-01 ± 1.581e-01
Пройденный путь	24.591 ± 6.159	36.675 ± 8.182	49.809 ± 9.141	62.039 ± 10.332
Количество итераций	133.334 ± 6.39	136.17 ± 4.261	137.772 ± 3.063	135.338 ± 3.241

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	5.099e-02 ± 7.492e-02	2.355e-02 ± 4.106e-02	1.309e-02 ± 2.583e-02
Пройденный путь	124.058 ± 14.270	186.119 ± 17.911	247.896 ± 20.727
Количество итераций	136.732 ± 1.754	137.077 ± 1.431	137.228 ± 1.309

Таблица А.14 – метрики алгоритма при радиусе связи, равном 1% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	3.153e-01 ± 2.524e-01	2.227e-01 ± 2.177e-01	1.817e-01 ± 1.894e-01	1.294e-01 ± 1.592e-01
Пройденный путь	24.503 ± 6.521	36.602 ± 7.951	49.132 ± 8.967	61.149 ± 10.145
Количество итераций	133.295 ± 6.513	136.424 ± 4.084	137.882 ± 3.263	135.315 ± 3.458

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	5.228e-02 ± 7.651e-02	2.647e-02 ± 4.321e-02	1.784e-02 ± 3.226e-02
Пройденный путь	121.882 ± 15.169	181.863 ± 18.823	241.866 ± 22.683
Количество итераций	136.79 ± 1.955	137.153 ± 1.722	137.404 ± 1.621

Таблица А.15 – метрики алгоритма при радиусе связи, равном 0.5% от размера поля, стратегия «edge»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	3.129e-01 ± 2.506e-01	2.193e-01 ± 2.141e-01	1.608e-01 ± 1.857e-01	1.349e-01 ± 1.620e-01
Пройденный путь	24.333 ± 6.407	37.088 ± 8.035	49.891 ± 9.254	62.065 ± 9.909
Количество итераций	132.612 ± 6.704	136.417 ± 3.869	137.474 ± 3.121	135.204 ± 3.411

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	5.091e-02 ± 7.955e–02	2.646e-02 ± 4.604e-02	1.606e-02 ± 3.135e-02
Пройденный путь	124.749 ± 14.777	186.780 ± 18.468	248.014 ± 20.674
Количество итераций	136.717 ± 1.747	136.976 ± 1.362	137.134 ± 1.161

Таблица А.16 – метрики алгоритма при радиусе связи, равном 0.5% от размера поля, стратегия «spot»

Метрика	Количество агентов			
	2 агента	3 агента	4 агента	5 агентов
Относительная ошибка, в долях	3.015e-01 ± 2.396e-01	2.284e-01 ± 2.167e-01	1.601e-01 ± 1.777e-01	1.375e-01 ± 1.576e-01
Пройденный путь	24.587 ± 6.417	36.992 ± 7.949	49.745 ± 9.476	62.068 ± 10.242
Количество итераций	133.204 ± 6.994	136.59 ± 4.043	137.751 ± 3.115	135.429 ± 3.371

Метрика	Количество агентов		
	10 агентов	15 агентов	20 агентов
Относительная ошибка, в долях	4.694e-02 ± 7.099e-02	2.837e-02 ± 4.942e-02	1.831e-02 ± 3.399e-02
Пройденный путь	122.972 ± 14.263	184.791 ± 17.453	247.254 ± 21.135
Количество итераций	136.742 ± 1.821	136.979 ± 1.486	137.367 ± 1.314

Приложение Б

(справочное)

QR-код на репозиторий

Этот QR-код приведёт вас в репозиторий с проектом, где вы сможете найти графики, не вошедшие в отчёт, а также весь исходный код.

Рисунок Б.1 – QR-код репозитория