УДК 004.896 DOI:10.17587/mau.18…

В.В. Воробьев, аспирант, gatus86@mail.ru,
Национальный исследовательский центр “Курчатовский институт”

Алгоритмы выбора лидера и кластеризации в статическом рое роботов[1]

Рассматриваются задачи выбора лидера и кластеризации в группе роботов. Показаны ряд подходов и методов к решению данных задач. Определены важные моменты, которые необходимо учитывать, решая эти задачи. Представлены алгоритмы выбора лидера и кластеризации, которые в разной степени учитывают данные моменты. Проведенные вычислительные эксперименты подтверждают работоспособность алгоритмов.

Ключевые слова: *групповая робототехника, выбор лидера, функциональная дифференциация, кластеризация, статический рой*

Введение

На сегодняшний момент робототехника является одним из приоритетных направлений исследований. Огромное внимание уделяется такой сфере, как групповая робототехника – обобщенное направление, куда входят роевая, стайная и коллективная робототехники [1]. Основным отличием их друг от друга является использование разных способов организации групп роботов, например, в соответствии со степенью информированности о целях, задачах, с учетом морфологии группы и т.д.[2]

Данный интерес вызван тем, что возможное практическое применение систем, состоящих из множества роботов, огромно: патрулирование [3], [4], разведка [5], диагностика труднодоступных объектов, работа в космосе [6] и т.д. Также, привлекательна появления возможность эмерджентных свойств, т.е. свойств, которыми не обладает ни одна из составных частей такой системы.

Одной из фундаментальных задач групповой робототехники является задача выбора лидера в группе роботов [7]. Действительно, существует ряд робототехнических задач, где наличие в группе лидера необходимо. Например, это задачи, которые решаются методами стайной робототехники: от примитивного движения за вожаком, до реализации подражательного

[1] Работа выполнена при частичной финансовой поддержке грантов РФФИ 16-29-04412 офи_м и РНФ 16-11-00018

поведения в целом, которое ведет к появлению сложных комплексов действий, например, строительство, уход, оборонительные действия и т.д. [8].

Решение подобной задачи описывается в [9], где представлены сразу несколько стратегий выбора лидера, которые, однако, подходят более для вычислительных сетей, нежели для группы роботов. В работе рассматриваются различные топологии сети и алгоритмы выбора лидера для них. Кроме того, представлен и ряд “универсальных” алгоритмов, наиболее интересный из которых YO-YO. Принцип его работы заключается в обмене заранее заданными весами между вычислительными устройствами. Лидером становится устройство с наименьшим весом.

Другой подход к выбору лидера, который разрабатывался специально для групп роботов, описан в [10]. В рассматриваемой модели роботы не имеют общей координатной системы и уникальных идентификаторов, группировка гомогенна. При этом роботы не могут опираться на информацию о вычислениях и сенсорных данных, сделанных ранее. Выбор лидера происходит по принципу близости робота к центру определенной формации, т.е. роботы сначала формируют некий паттерн, который одновременно является чем-то вроде глобальной системы координат, а затем определяются финальные позиции каждого робота. В конце они стараются достичь данных позиций. Тот робот, который окажется ближе всего к центру описанной окружности данной формации и станет лидером.

Еще один механизм выбора лидера описан в [1], суть которого, заключается в том, что робот определяет за кого проголосовали его соседи. В зависимости от веса кандидата, за которого голосует его сосед, робот может поменять свой выбор и проголосовать за того же кандидата. Особенностью такого подхода является то, что нет необходимости в ведении уникальных идентификаторов роботов, а также то, что они используют только локальное взаимодействие друг с другом. Сам процесс выбора лидера происходит в структуре, называемой статическим роем – некой фиксированной в

определенный момент времени сети, состоящая из роботов, соединенных друг с другом по каналам связи [11].

Как уже было сказано, для решения некоторых задач стайной робототехники необходимо наличие сложных комплексов действий, например, для задачи группового патрулирования. В свою очередь это предполагает наличие механизма функциональной дифференциации, т.е. распределения ролей в группе роботов. Эту задачу можно свести к задаче кластеризации группы роботов, где каждому кластеру отведена своя роль в выполнении общей задачи.

Механизм кластеризации, реализованный в работах [12],[13], основан на гранулярной конвекции или на эффекте “бразильского ореха”. Основная идея работы – группировать роботов по принципу случайного движения вокруг общей точки притяжения и отталкивания между соседями.

В работе [14], которая является идеологическим продолжением работы [15], в основе алгоритма кластеризации, который содержит в себе элементы, используемые в бисекции и балансировки нагрузки в сетях, лежит идея обмена жетонами (token) между роботами. Наличие жетона определяет принадлежность к кластеру.

Подробно описывается процесс кластеризации в работе [2]. Выделяется иерархическая кластеризация, причем иерархия может выстраиваться двумя путями: “сверху-вниз” и “снизу-вверх. Кроме того, предложены алгоритмы создания непересекающихся кластеров постоянного состава, как для гетерогенных, так и для гомогенных групп роботов. Рассматриваются проблемы перераспределения роботов между кластерами и динамической кластеризации – процесс инициализации кластеров и их роста.

Отдельно стоит упомянуть работу [1], где решается задача распределения ролей без кластеризации. Здесь роли распределяются в соответствии с топологическим расстоянием от лидера.

Специфика рассмотренных методов заключается в том, что лидером может стать любой робот, в том числе и периферийный, т.е. тот, кто находится на

краю группы. В условиях, когда роботы общаются исключительно локально, т.е. только со своими соседями, может возникнуть ситуация, когда время обмена данными с лидером существенно увеличивается. Поэтому важно чтобы лидером становился робот, который находится близко к центру группы. Другими словами специфика решения такой задачи должно учитывать и топологические характеристики группировки, т.е. их взаимное расположение.

1. Постановка задачи

Предположим, что есть некоторая гомогенная группа роботов, представленная в виде статического роя – фиксированной в определенный момент времени структуры, состоящей из роботов, соединенных друг с другом по каналам связи [11]. Каждый робот имеет определенное количество каналов связи, при этом связь носит исключительно локальный характер. Пример реализации такой связи, например, описан в работе [16]. Безусловно, это несколько ограничивает возможности обмена данными, однако позволяет работать в таком случае с большим количеством роботов, т.к. реализация связи “все-со-всеми” трудна или полностью невозможна в случае большой численности (от нескольких десятков и более роботов).

По наступлению заранее заданного события, например, сигнала извне, группа собирается в статический рой [11]. При этом ни одному роботу не известна ни численность группировки, ни топология статического роя. Необходимо гарантированно перевести группу роботов из состояния статического роя без лидера в состояние, где один из них лидером является. Обязательным условием является то, что лидер должен находиться как можно ближе к центру роя, т.к. это может минимизировать время обмена данными между роботами на периферии статического роя и лидером. Сам механизм обмена основан на ретрансляции сообщений с помощью соседей.

Более формально задачу выбора лидера можно описать следующим образом: существует рой роботов, числом N, где каждый робот описывается пятеркой:

$$V_i=(\alpha,L,C,W,W_{my})$$

здесь α – уникальный идентификатор V, L – список соседей V, C – идентификатор робота-лидера, за которую голосует V, W – вес кандидата C, W_{my} – собственный вес V. Это похоже на формализацию, предложенную в работе [17] за тем исключением, что введен еще один атрибут W_{my}, который необходим для случая переголосования. Необходимо гарантированно перевести рой из начального состояния:

$$S_{swarm}=\{V_1,V_2,\ldots,V_N\},\ \text{где для любого}\ V_i$$

$$C_i=\varnothing,\ i=1,2,\ldots,N$$

в состояние:

$$S_{flock}=\{V_1,V_2,\ldots,V_N\},\ \text{где существует единственный}\ V_i,\ \text{такой что}$$

$$C_i=\alpha_i,\ i=1,2,\ldots,N\ \text{и где для любого}\ V_j$$

$$C_j=\alpha_i,\ j=1,2,\ldots,N;\ j\neq i;$$

Гарантированность процесса можно описать следующим образом:

$|S|=k+2$ для любого N, где $S=\{S_{swarm},S_1,\ldots,S_k,S_{flock}\}$ – множество состояний роя при переходе из S_{swarm} в S_{flock}.

Если интерпретировать рой как граф G, где роботы являются вершинами, а связь с соседями - ребрами, то близость лидера к центру роя определяется следующим образом:

$$d = \frac{s}{r(G)}$$

где r – радиус графа G, s – расстояние от центра графа до робота-лидера.

После выбора лидера и выработки конкретных планов действий необходимо произвести функциональную дифференциацию, следовательно, группа должна разделиться на M заданных кластеров. Таким образом, необходимо разделить граф G на M подграфов:

$$\text{Существует}\ f_{div}(G)=\{G_1,G_2,\ldots,G_M\},\ \text{такое что}$$

$$g(f_{div})=T_{div}\leq const$$

где f_{div} – функция разделения графа G на M подграфов, $g(\text{f}_{\text{div}})$ – функция, возвращающая время выполнения f_{div}, T_{div} – время деления графа G на M подграфов.

2. Алгоритм выбора лидера

Статический рой роботов в начале работы алгоритма выбора лидера находится в состоянии S_{swarm}, которое характеризуется тем, что ни один робот не является лидером. У каждого робота V вес $W_{\text{my}}=1$, однако, ни один робот еще не выбирал лидера, поэтому $C_i=\varnothing$ и $W=0$. При этом отдельные роботы, которые находятся на периферии статического роя, могут иметь свободные каналы связи, т.е. длина списка L_i соседей V_i меньше максимальной длины данного списка $len_{\max}$. Технически это означает, что некоторые из каналов локальной связи робота “не видят” соседей. Такие роботы считаются периферийными, и именно они инициируют процесс выбора лидера.

Сам алгоритм выбора лидера можно условно разделить на три стадии: определение весов кандидатов, отбраковка кандидатов, после чего, остается только один кандидат, и подтверждение лидерства, необходимое для того, чтобы сообщить кандидату, что весь рой согласен с его лидерством. Частично данный механизм описан в работе [17].

2.1. Выявление кандидатов

Процедура выявления кандидатов на лидерство (Алгоритм 1) начинается с того, что периферийные роботы, т.е. роботы, число соседей $|L|$ которых меньше максимальной длины данного списка $len_{\max}$:

$$V_i, \text{ такой что } |L_i|<len_{\max}, \; i=1,2,\ldots,N$$

посылают сообщение с весом W_{my} всему списку своих соседей L. Если сосед не периферийный робот, то он принимает и анализирует данное сообщение и прибавляет полученный W_{my} к своему W_{Lmy}. Анализ необходим для того чтобы избежать ситуации, когда вес W_{my} прибавляется дважды. Затем роботы, получившие это сообщение, также посылают свой вес W_{Lmy} своим соседям, инициируя процедуру формирования их веса и т.д. Когда робот

сформировал свой вес, т.е. обработал все сообщения от своих соседей, он выдвигает свою кандидатуру на лидерство.

Благодаря такому механизму роботы, которые ближе к центру статического роя будут иметь большие веса, нежели те, которые на периферии. Следовательно, это позволяет избегать ситуаций, когда лидером становится периферийный робот.

***Алгоритм 1**. Определение веса кандидата*

1. Проверка, является ли робот *V* периферийным. Если это так, то шаг 5. Иначе, шаг 2.
2. Если буфер сообщений робота *V* не пуст и сообщение хранит вес соседа, то шаг 3. Иначе, если все сообщения обработаны, то шаг 5. Иначе, продолжать ожидание.
3. Проверка, получал ли уже робот *V* сообщение с весом от этого соседа. Если это так, то шаг 2. Иначе, шаг 4.
4. V прибавляет полученный вес к своему. $W_{\text{Lmy}}:=W_{\text{Lmy}}+W_{\text{my}}$. Шаг 2.
5. Отправить свой вес всем соседям, кроме тех, от кого получил сообщение с весом. Периферийные роботы отправляют сообщение с весом всем своим соседям. Шаг 6.
6. V выдвигает свою кандидатуру на лидерство.

2.2. Выбор кандидата с наибольшим весом

Как только робот становится кандидатом на лидерство, т.е. его $C=\alpha$, он сообщает об этом своим соседям (Алгоритм 2) специальным сообщением. Если сосед L_i сам не является кандидатом, то он считает лидером робота-кандидата, запоминает его вес $W_i=W$, при этом, не меняя свой собственный вес W_{my}. После этого он передает информацию о лидере всем своим соседям, кроме тех соседей, которые ему данную информацию передали, т.е. кроме “родительских” роботов. Это позволяет выстраивать иерархию связей, где все потоки данных направлены от периферии к лидеру, т.е. от “потомков” к “родителям”, а все потоки команд от лидера к периферии, т.е. от “родителей” к “потомкам”.

Роботы-соседи также запоминают вес W. Если возникает ситуация, когда принимающий робот является кандидатом или уже считает лидером другого робота, то сравниваются веса этих лидеров W. Робот с меньшим весом лидера присоединяется к роботу с большим весом. Если веса одинаковы, то лидерство определяется по уникальному идентификатору робота-кандидата α – главным становится робот с большим его значением. Это необходимо для того, чтобы избежать ситуаций с бесконечным выбором лидера, которые описаны в работе [1].

Таким образом, в рое остается только один лидер, который, однако, еще не знает об этом. Это связано с тем фактом, что лидеру не известна численность и топология роя, поэтому невозможно гарантировать то, что этот лидер единственный на данный момент. В связи с этим необходима процедура подтверждения, для того чтобы гарантировать наличие только одного лидера.

Алгоритм 2. *Отбраковка кандидатов*

1. Проверка, является ли робот *V* кандидатом. Если это так, то шаг 2. Иначе, шаг 4.
2. Сформировать вес кандидата $W=W_{my}$. Шаг 3.
3. Передать всем соседям вес кандидата *W*. Шаг 4.
4. Если буфер сообщений робота *V* не пуст, и сообщение хранит данные о кандидате то шаг 5. Иначе, шаг 6.
5. Если вес кандидата W_i, за которого голосует робот V **меньше**, чем вес кандидата в сообщении $\boldsymbol{W_i<W,}$ или если они равны $W_i == W$, но идентификатор α_i робота-кандидата **меньше**, чем идентификатор кандидата в сообщении $\boldsymbol{\alpha_i<\alpha}$, то *V* голосует за кандидата в сообщении. Если он при этом сам был кандидатом, то он перестает им быть. Шаг 4.
6. *V* передает вес и идентификатор кандидата, за кого он голосует всем соседям, кроме тех, от кого он получил сообщение о данной кандидатуре.

2.3. Подтверждение лидерства

Подтверждение лидерства происходит параллельно с предыдущим этапом: любой кандидат, заявляя о своем лидерстве соседям, ждет согласие от них (Алгоритм 3). Робот соглашается с лидерством, если согласны с данным лидером все его соседи, кроме родительских. В случае если у робота есть только родители, то он соглашается с лидерством кандидата, за которого он голосует. Фактически сначала распространяется опрашивающая волна от лидера до периферии роя, а затем волна-ответ идет от периферии к лидеру.

Алгоритм 3. *Подтверждение лидера*

1. Если у робота V нет соседей, кому он может сообщить о кандидате, то шаг 2. Иначе, шаг 3.
2. Передать согласие с лидерством родителям. Шаг 6.
3. Если буфер сообщений робота V не пуст, и сообщение хранит согласие от “потомка”, то шаг 4. Иначе, шаг 6.
4. Увеличить число согласившихся потомков на 1. Шаг 5.
5. Если все потомки согласились с лидерством, то V также соглашается с лидерством и сообщает об этом своим родителям. Иначе, шаг 3.
6. Ожидание сообщений.

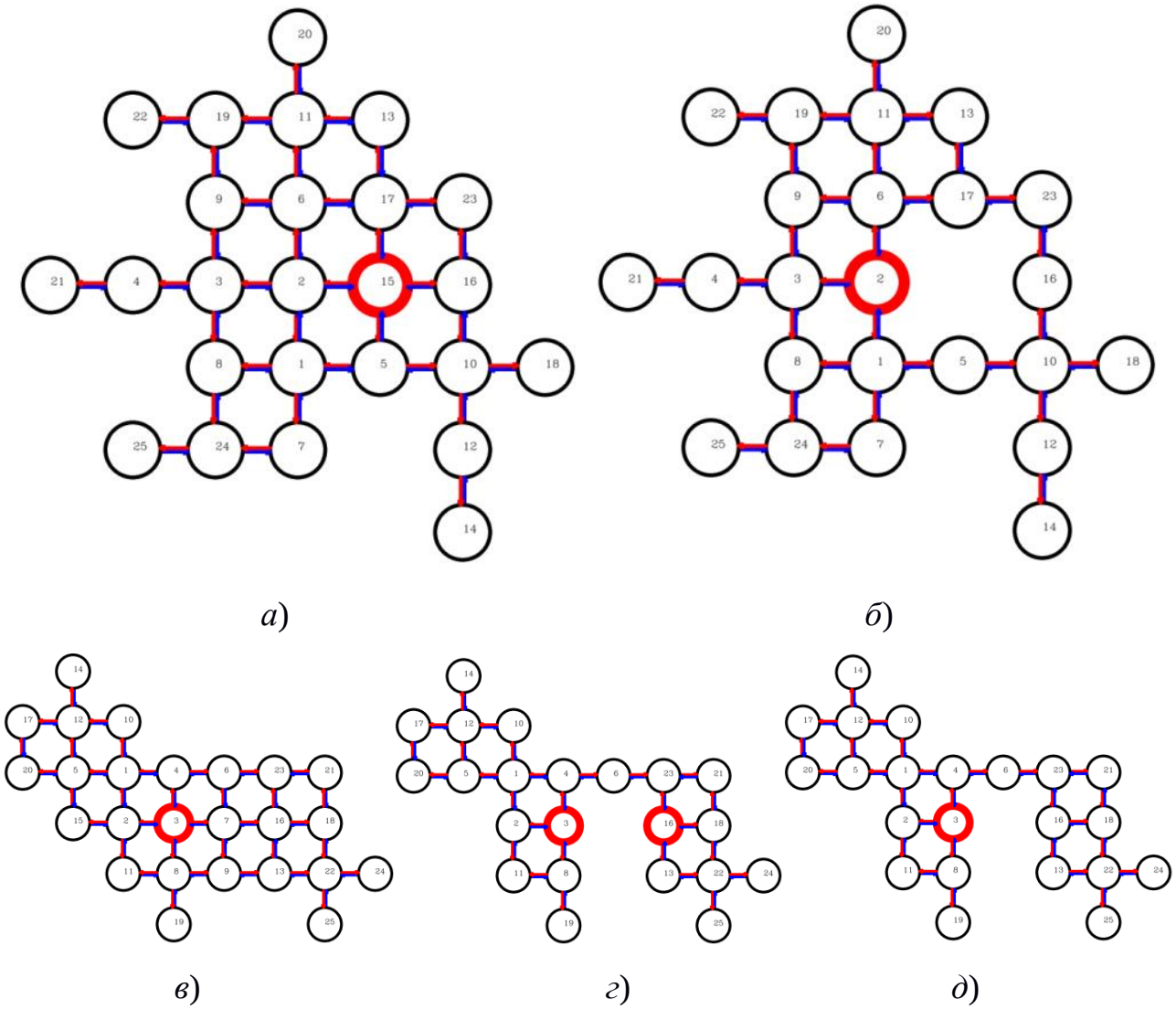
2.4. Потеря лидера

В основе механизма перевыборов лидера лежит правило: если у робота нет “родителя”, т.е. ему некуда передавать данные и ему никто не присылает команды, и он не является лидером, то он выдвигает свою кандидатуру на лидерство.

Например, есть рой с N=25 и максимальной длиной $|L|$=4 (Рис. 1,*а*, Рис. 1,*в*) и из-за технических неполадок или сознательного подавления каналов связи извне, робот теряет своего “родителя”. Технически это реализовано с помощью таймера, который обнуляется, когда от “родителя” приходит любое сообщение. По его переполнению “потомок” выдвигает свою кандидатуру на

нового лидера. Затем повторяется уже известная процедура выбора кандидата с наибольшим весом и подтверждение кандидатуры (Рис. 1,*б*).

Аналогичная ситуация наблюдается если лидер остался, но теряется связь с роботами, которые ретранслируют данные от периферийных роботов. Те, кто потерял “родителей” также выдвинут свою кандидатуру (Рис. 1,*г*), но когда связь с лидером восстановится по альтернативным каналам, они снова проголосуют за старого лидера (Рис. 1,*д*).



а) *б*) *в*) *г*) *д*)

Рис. 1. Переголосование. Толстой линией выделен лидер

3. Задача кластеризации

Суть задачи заключается в необходимости функциональной дифференциации ролей в группе, т.е. лидер, определив необходимые роли, распределяет их между членами группы. Решение этой задачи осуществляется

поэтапно: лидер формирует зачатки кластера и, при необходимости, передает информацию о дальнейшей кластеризации. Затем в каждом кластере выбирается свой лидер, который уже имеет информацию о том, надо ли делиться дальше. Если такая необходимость есть, то процесс повторяется [18].

Непосредственно алгоритм кластеризации представляет собой несколько процедур деления/выбора лидера. Их число зависит от количества кластеров, на которые необходимо поделить рой и числа соседей у робота-лидера, которым он может отправить сообщение о начале кластеризации.

Старт процедуры инициируется сообщением, которое отправляет робот-лидер своим соседям. При этом если число соседей больше или равно числу кластеров, на которые рой необходимо разделить, то робот-лидер произвольно выбирает, кому он отправит данное сообщение. Каждому из выбранных соседей сообщается уникальный номер кластера. В свою очередь эти роботы отправляют такое же сообщение с уникальным номером кластера своим соседям и т.д. Если робот уже принадлежит какому-либо кластеру, то он не может принимать сообщения от частей с другим уникальным номером, что позволяет закончить работу алгоритма разделением роя на заданное количество частей.

В случае если разделить группу необходимо на части, количество которых больше, чем количество соседей робота-лидера, то выполнение алгоритма начинается так же, как если бы они были равны – лидер делит группу на число кластеров, которое равно числу его соседей. При этом в сообщении о разделении вместе с номером формирующегося кластера хранится информация о том, на сколько частей его необходимо разделить в дальнейшем. После того, как рой разделится первый раз, в кластерах инициируется процедура выбора лидера, который вновь запустит кластеризацию на то количество частей, которое ему было передано во время первого этапа кластеризации.

Так как неизвестно заранее, какой робот в кластере станет лидером, то информация об этом передается всем роботам в процессе кластеризации.

Лидер в кластере выбирается так же, как это описано в разделе 2. Важной деталью является то, что этот процесс синхронизирован во всем кластере, т.е. все роботы, относящиеся к данному кластеру, начинают процесс выбора лидера одновременно. Это возможно благодаря тому, что роботы могут “впадать в спячку“ на заданное время. Время определяется следующим образом: на этапе выбора лидера (раздел 2.2) голосующие за лидера роботы определяют количество роботов между ними и лидером, другими словами, определяют расстояние до него. Процесс определения расстояния заключается в следующем: лидер имеет расстояние RL=0 и передает его своим соседям. Они формируют свое расстояние RN=RL+1 и передают своим соседям (исключая “родителя”) и т.д. до периферийных роботов. Когда осуществляется подтверждение лидерства (раздел 2.3) вместе с подтверждением роботы отправляют свое расстояние до лидера. Лидер выбирает максимальное расстояние R_{max}. Когда начинается процесс кластеризации, лидер передает R_{max} всем роботам группы, каждый из которых определяет свое ожидание, исходя из собственного расстояния до лидера:

$$T_i=(R_{max}-R_i)k$$

где T_i – время ожидания, k – величина, обратная скорости движения сообщения от лидера до периферии, R_{max} – максимальное расстояние от лидера до периферии, R_i – расстояние робота i до лидера. Определив свою принадлежность к какому-либо кластеру, робот ждет на протяжении времени T_i, а затем начинает процесс выбора лидера.

Алгоритм 4. *Алгоритм кластеризации*

1. Если робот V – лидер, то шаг 2. Иначе шаг 3.
2. Определение числа необходимых делений. Передача соседям номеров кластеров, к которым они принадлежат и числа последующих делений каждого кластера. Снятие с себя функции лидера. Шаг 3.
3. Если буфер сообщений робота V не пуст, и сообщение хранит в

себе номер кластера и число последующих делений, то шаг 4. Иначе шаг 3.

4. *V* присваивает себе полученный номер кластера и число делений, которые находятся в сообщении и передает этот номер и число делений дальше, своим соседям, кроме тех, чей номер кластера совпадает с номером кластера *V*. Вычисление времени ожидания *T*. Шаг 5.
5. Ожидание в течение времени *T*.

4.Вычислительные эксперименты

Было проведено имитационное моделирование, состоящее из серии вычислительных экспериментов. Входными параметрами модели являлись количество агентов N и максимальное число каналов связи агента K, т.е. максимальное число его соседей. Агенты располагались случайным образом. Для каждого N (N=50,75,100…200), было проведено сто вычислений. Другими словами для каждого N определялось среднее значение времени T_{map}, которое необходимо для того, чтобы перевести систему из состояния S_{swarm} в состояние S_{flock}. Кроме того, вычислялось среднее значение отклонения лидера от центра роя $d = \frac{s}{r(G)}$, проверялась работоспособность функции $f_{\text{div}}(G)= \{G_1,G_2,\ldots,G_M\}$ и определялось T_{div} – время ее выполнения. Также вычислялись средние размеры каждого кластера $G_1,G_2,\ldots,G_M$.

Рис. 2 демонстрирует среднее время работы T_{map} в зависимости от количества роботов N для случая K=4, а Рис. 3 - для K=8. T_{map} измеряется в тактах работы алгоритма, каждый из которых состоит из операций приема данных от соседей, их обработки и передачи этих данных соседям. Ромбами отмечается время выполнения первого этапа работы алгоритма, квадратами – второго, треугольниками – третьего. Временная сложность оценивается как O(N).

Ни один вычислительный эксперимент не закончился тем, что лидер выбирался бесконечно, что говорит о некоей статистической достоверности того, что лидер будет гарантированно выбран.

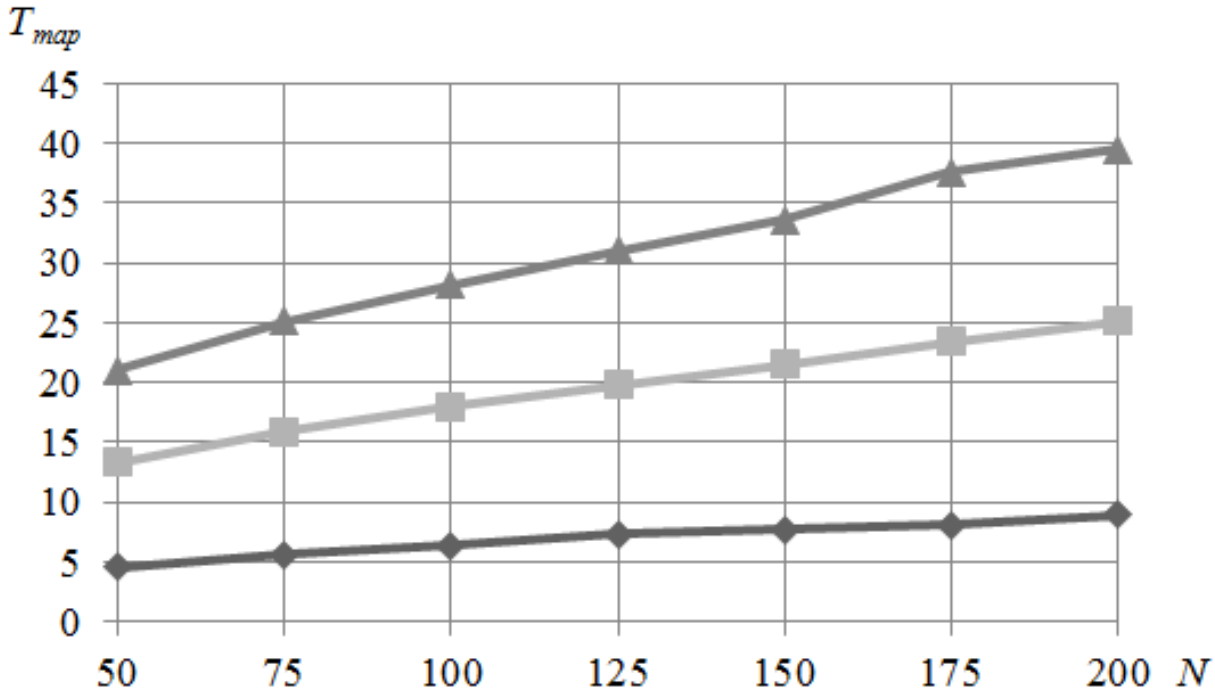


Рис. 2. Зависимость времени выбора лидера от количества роботов для *K*=4

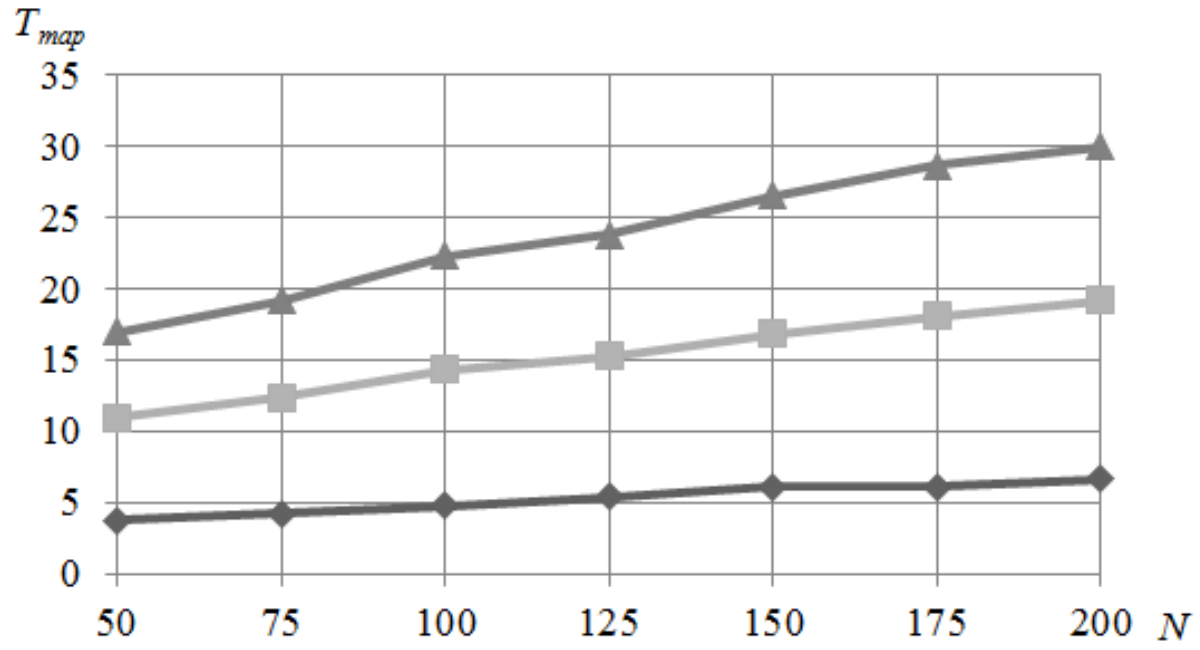


Рис. 3. Зависимость времени выбора лидера от количества роботов для *K*=8

На Рис. 5показано среднее значение величины отклонения *d* от центра роя для случая *K*=4, а на Рис. 5 для K=8. В среднем, отклонение от центра не более

0,25 длины радиуса роя. Это подтверждает то, что алгоритм учитывает требование того, чтобы лидер был близко к центру группы роботов.

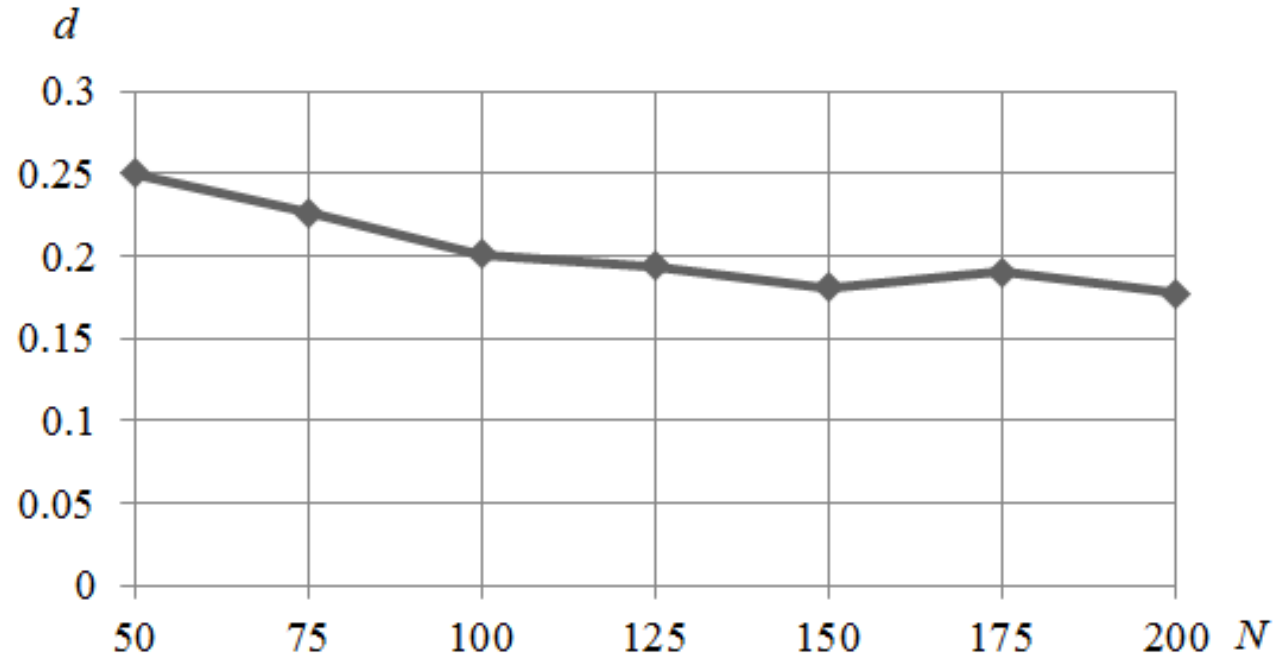


Рис. 4. Среднее отклонение от центра роя *d* для *K*=4

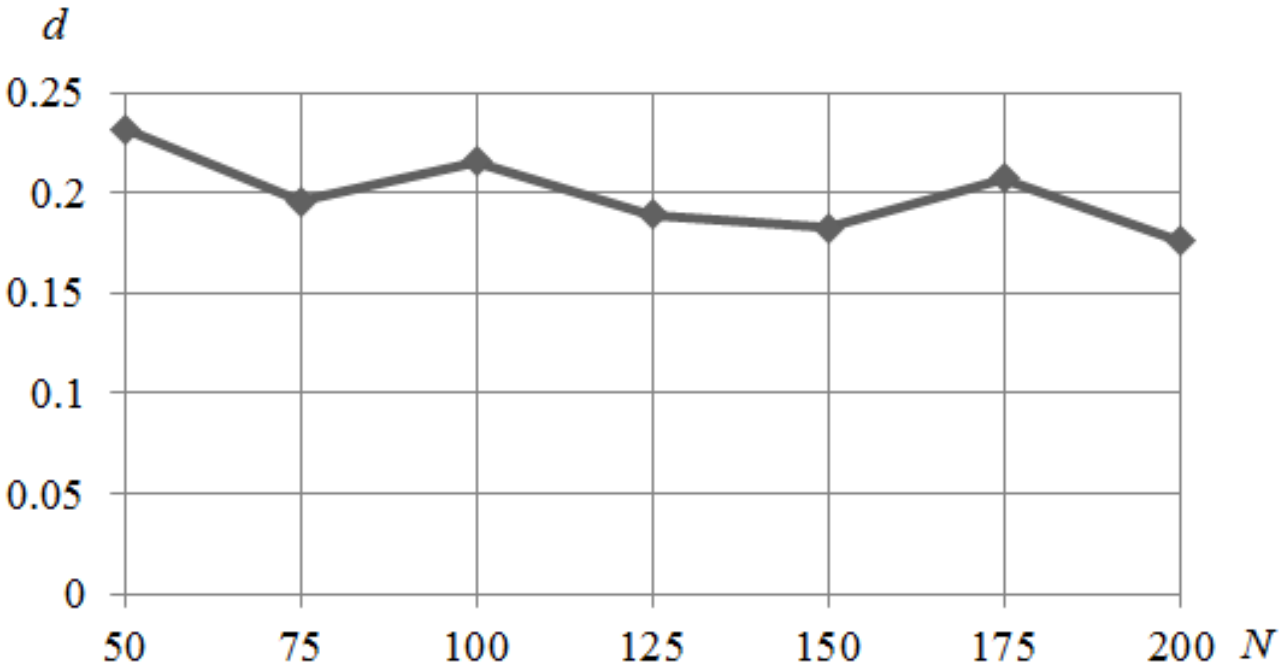


Рис. 5. Среднее отклонение от центра роя *d* для *K*=8

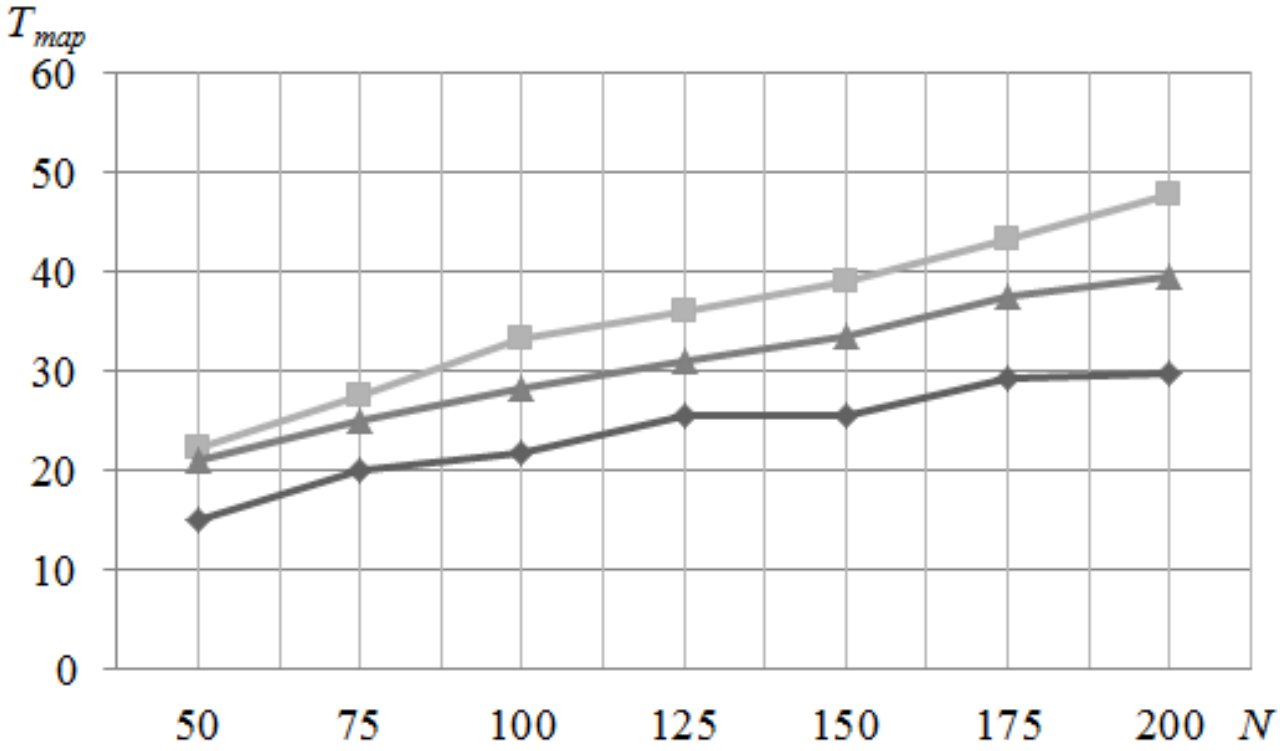


Рис. 6. Выбор лидера с выбыванием

Кроме того промоделирована ситуация для числа каналов связи у робота K=4, когда в процессе выбора лидера часть из них выбывает из роя, т.е. нарушается топология связей (Рис. 6). На рисунке квадратами изображена кривая времени, при выбывании 25% роя, ромбами – 50%, а треугольниками – эталон без выбывания. Роботы начинают выбывать сразу после начала процедуры выборов в течение 5 тактов. Из графика видно, что выбывание 50% роя ведет к существенному упрощению топологии роя, что приводит к уменьшению времени выбора. С ситуацией выбывания 25% дело обстоит иначе – выбывших узлов не достаточно для того, чтобы существенно упростить топологию; наоборот, она усложняется, что ведет увеличению T_{map}.

На Рис. 7 изображено время кластеризации T_{div} для случая K=4. В T_{div} также входит время выбора лидера в новом кластере.

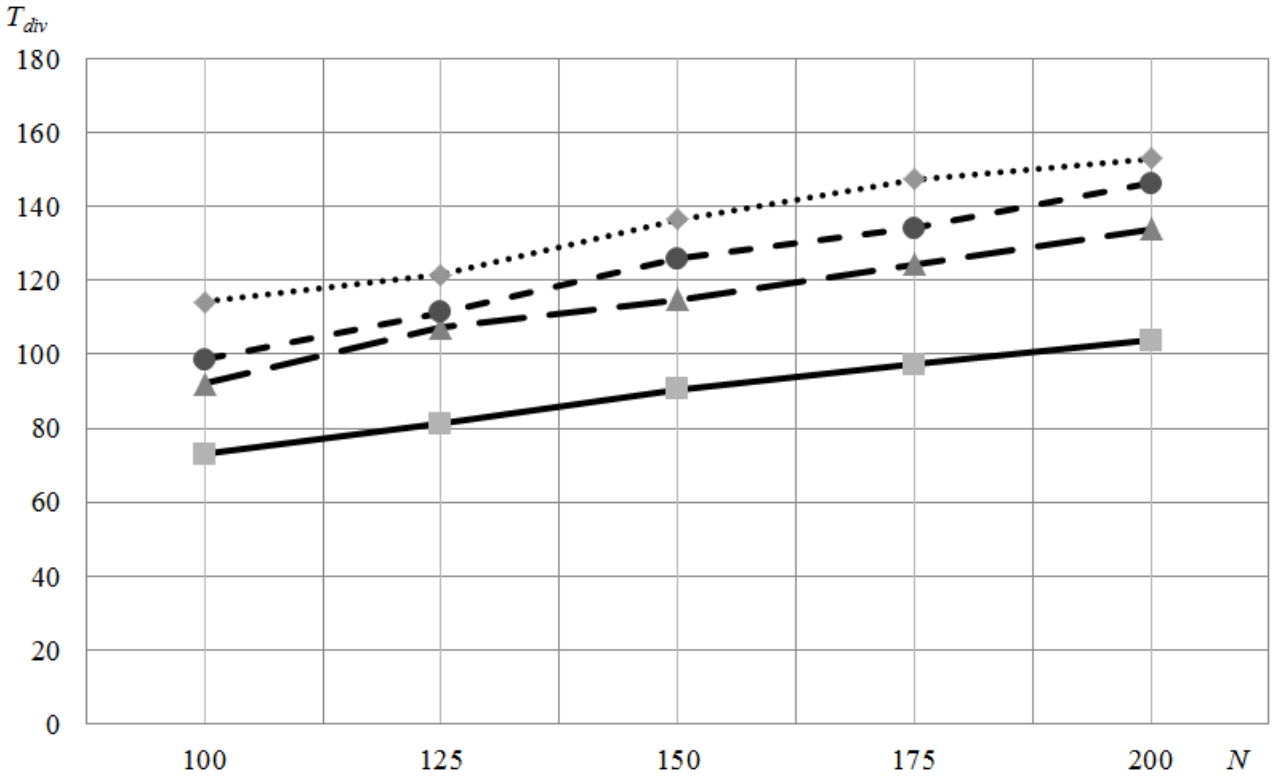


Рис. 7. Время кластеризации: сплошная линия – 4 кластера, длинная штриховая – 6 кластеров, короткая штриховая – 8 кластеров, пунктирная линия – 10 кластеров

Размер каждого кластера в процентах от общего размера группы, для случая, когда кластеров шесть, показан на Рис. 8.

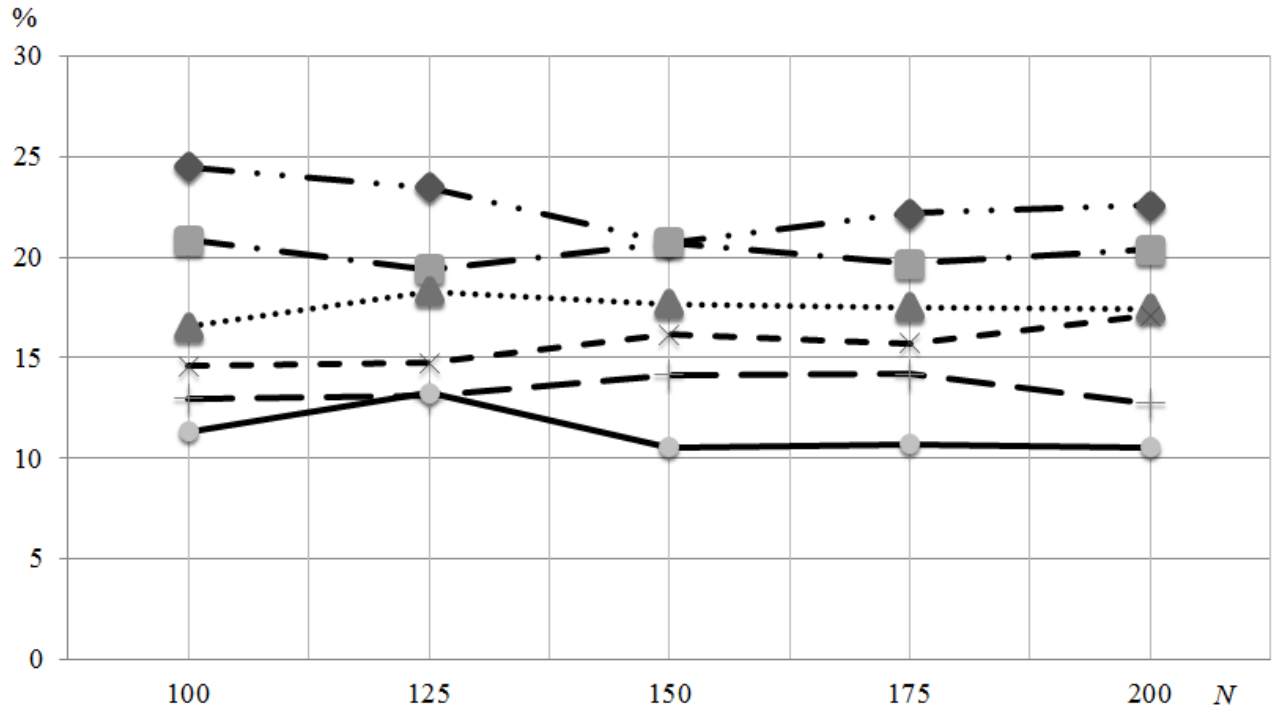


Рис. 8. Размер каждого кластера (число кластеров = 6)

Заключение

В работе были предложены алгоритмы выбора лидера к группе роботов и кластеризации данной группы, которые работают на модели статического роя [11]. В отличие от существующих алгоритмов выбора лидера предлагаемый алгоритм позволяет не просто выбирать лидера, но и учитывать топологию группы где он выбирается, что ведет к тому, что лидером могут стать только те роботы, которые находятся близко к центру. Кроме того, алгоритм успешно завершает работу и в случаях, когда нарушается топология связей в группе. Если сравнивать время выбора лидера, то алгоритм часто показывает результаты, аналогичные результатам, представленным в работе [1].

Алгоритм кластеризации, необходимый для функциональной дифференциации роботов также показал стабильную работу, все вычислительные эксперименты завершились удачным разделением на кластеры за конечное время T_{div}.

Таким образом, для модели статического роя предложен механизм, для решения задач выбора лидера и функциональной дифференциации с учетом топологии связей в группе.

Список литературы

1. **Karpov V., Karpova I.** Leader election algorithms for static swarms // Biologically Inspired Cognitive Architectures. 2015. №12. С.54-64
2. **Каляев И. А., Гайдук А. Р., Капустян С. Г.** Модели и алгоритмы коллективного управления в группах роботов. М.: Физматлит, 2009. 280 с.
3. **Bina D.** Effective cooperation and scalability in multi-robot teams for automatic patrolling of infrastructures // Coimbra. 2013. 246 с.
4. **Portugal D., Rocha R. P.** Cooperative multi-robot patrol with Bayesian learning //Autonomous Robots. 2016. Т. 40. №. 5. С. 929-953.
5. **Tan Y., Zheng Z.** Research advance in swarm robotics //Defence Technology. 2013. Т. 9. №. 1. С. 18-39.

6. **Canepa D., Potop-Butucaru M. G.** Stabilizing flocking via leader election in robot networks //Symposium on Self-Stabilizing Systems. Springer Berlin Heidelberg, 2007. С. 52-66.
7. **Dieudonné Y., Petit F., Villain V.** Leader election problem versus pattern formation problem // International Symposium on Distributed Computing. Springer Berlin Heidelberg, 2010. С. 267-281.
8. **Карпов В. Э.** Коллективное поведение роботов. Желаемое и действительное // Современная мехатроника. Сб. научн. трудов Всероссийской научной школы (22-23 сентября 2011г., г. Орехово-Зуево, Россия): Труды конференции, 2011. С. 35–51.
9. **Santoro N.** Design and analysis of distributed algorithms. John Wiley & Sons, 2006. 589 с.
10. **Chaudhuri S. G., Mukhopadhyaya K.** Leader election and gathering for asynchronous fat robots without common chirality //J ournal of Discrete Algorithms. 2015. Т. 33. С. 171-192.
11. **Карпов В.Э.** Управление в статических роях. Постановка задачи // VII-я Международная научно-практическая конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте» (20-22 мая 2013, г. Коломна, Россия): Труды конференции. В 3-х томах. Т.2. М.: Физматлит, 2013. С.730 –739.
12. **Groß R., Magnenat S., Mondada F.** Segregation in swarms of mobile robots based on the Brazil nut effect //2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009. С. 4349-4356.
13. **Chen J**. et al. Segregation in swarms of e-puck robots based on the brazil nut effect // Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems. 2012. Т.1. С. 163-170.
14. **Cruz N. B., Nedjah N., de Macedo Mourelle L.** Robust distributed spatial clustering for swarm robotic based systems //Applied Soft Computing. 2016.
15. **Di Caro G. A., Ducatelle F., Gambardella L.** A fully distributed communication-based approach for spatial clustering in robotic swarms // Proc.

of the 2nd Autonomous Robots and Multirobot Systems Workshop (ARMS), affiliated with the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)(Valencia, Spain, June 5). 2012. С. 153-171.

16. **Карпова И.П.** Псевдоаналоговая коммуникация в группе роботов //М: Мехатроника, автоматизация, управление. 2016. Т. 17. №2. С. 94 - 101.
17. **Воробьев В.В., Московский А.Д.** Алгоритм выбора лидера в системах с меняющейся топологией // Пятнадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2016 (3-7 октября 2016, г. Смоленск, Россия): Труды конференции. В 3-х томах. Т.1. Смоленск: Универсум, 2016. С. 149–157.
18. **Воробьев В.В.** Алгоритм кластеризации коллектива роботов // Третий Всероссийский научно-практический семинар "Беспилотные транспортные средства с элементами искуственного интеллекта" (БТС-ИИ-2016, 22-23 сентября 2016 г., г. Иннополис, Республика Татарстан, Россия): Труды семинара. М: Издательство "Перо", 2016. С. 50–59.

Leader Selection and Clusterization Algorithms in a Static Robot Swarm

V.V.Vorobyov, gatus86@mail.ru,

Moscow Institute of Physics and Technology,
Moscow, 141700, Russian Federation

Corresponding author:
Vorobyov V.V., Postgraduate Student, Moscow Institute of Physics and Technology, Moscow, 141700, Russian Federation
e-mail: gatus86@mail.ru

Received on November 24, 2016
Accepted on November 30, 2016

The paper presents the problem of clustering and leader selection in a group of robots, using a static swarm model - fixed network at some point in time, consisting of the robots connected to one another via the communication channels. Robots use only local interaction, the topology of the swarm and the number of the robots is not known beforehand. It is proposed to take into account the relative positions of the robots and their neighbors, i.e., their local topology, which is known to them, and allows them, in the long run, to choose their leader out of the robots located close enough to the topological center of the whole group. It is known that the group has peripheral robots - those which have not all the communication channels occupied. They initiate the leader selection procedure by transmitting its weight to the center of the group. This allows them to create there a subgroup of robots, with the biggest weights, one of which becomes the leader. It is considered as an option, when the static topology of the swarm changes, i.e. some robots are eliminated in the process of voting. It is demonstrated that in all these cases the leader selection algorithm succeeds. In addition, a clustering algorithm is proposed intended to solve the problem of the functional differentiation of robots, which will quickly produce their integration into subgroups. The conducted computing experiments prove the efficiency of the algorithms.

Keywords: *swarm robotics, leader selection, functional differentiation, clustering, static swarm, flocking*

Acknowledgements: The work was supported by grants of RNF 16-11-00018 and RFBR 16-29-04412 ofi_m

For citation:
Vorobev V. V. Leader Selection and Clusterization Algorithms in a Static Robot Swarm, *Mekhatronika, Avtomatizatsiya, Upravlenie,* 2017, vol. 18, no. 3, pp. …
DOI: 10.17587/mau.18.....

References

1. **Karpov V., Karpova I.** *Biologically Inspired Cognitive Architectures*, 2015, vol. 12, pp. 54–64.
2. **Kaljaev I. A., Gajduk A. R., Kapustjan S. G.** *Modeli i algoritmy kollektivnogo upravlenija v gruppah robotov* (…), Moscow, Fizmatlit, 2009, 280 p.

3. **Bina D.** Effective cooperation and scalability in multi-robot teams for automatic patrolling of infrastructures // Coimbra. – 2013. – 246 p.
4. **Portugal D., Rocha R. P.** *Autonomous Robots*, 2016, vol. 40, iss. 5, pp. 929-953.
5. **Tan Y., Zheng Z.** *Defence Technology*, 2013, vol. 9, iss. 1, pp. 18-39.
6. **Canepa D., Potop-Butucaru M. G.** *Symposium on Self-Stabilizing Systems*, Springer Berlin Heidelberg, 2007, pp. 52-66.
7. **Dieudonné Y., Petit F., Villain V.** *International Symposium on Distributed Computing*, Springer Berlin Heidelberg, 2010, pp. 267-281.
8. **Karpov V. Je.** *Sovremennaja mehatronika, Sb. nauchn. trudov Vserossijskoj nauchnoj shkoly* (22-23 sentjabrja 2011g., g. Orehovo-Zuevo, Rossija, 2011, pp. 35–51.
9. **Santoro N.** Design and analysis of distributed algorithms, John Wiley & Sons, 2006, 589 p.
10. **Chaudhuri S. G., Mukhopadhyaya K.** *Journal of Discrete Algorithms*, 2015, vol. 33, pp. 171-192.
11. **Karpov V.Je.** *VII International scientific-practical conference «Integrirovannye modeli i mjagkie vychislenija v iskusstvennom intellekte»* (20-22 maja 2013, g. Kolomna, Rossija), vol.2, Moscow Fizmatlit, 2013, pp.730 – 739.
12. **Groß R., Magnenat S., Mondada F.** 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4349-4356.
13. **Chen J. et al.** *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, vol.1, pp. 163-170.
14. **Cruz N. B., Nedjah N., de Macedo Mourelle L.** *Applied Soft Computing*, 2016.
15. **Di Caro G. A., Ducatelle F., Gambardella L.** *Proc. of the 2nd Autonomous Robots and Multirobot Systems Workshop (ARMS), affiliated with the 11th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS), Valencia, Spain, June 5, 2012, pp. 153-171.
16. **Karpova I.P.** *Mehatronika, Avtomatizacija, Upravlenie*, 2016, vol. 17, iss. 2, pp. 94 – 101.
17. **Vorob'ev V.V., Moskovskij A.D.** *Pjatnadcataja nacional'naja konferencija po iskusstvennomu intellektu s mezhdunarodnym uchastiem KII-2016* (3-7 oktjabrja 2016, g. Smolensk, Rossija), vol.1, Smolensk, Universum, 2016, pp. 149–157.
18. **Vorob'ev V.V.** *Third All-Russian scientific-practical seminar "Bespilotnye transportnye sredstva s jelementami iskustvennogo intellekta"* (BTS-II-2016, 22-23 sentjabrja 2016, Innopolis, Respublika Tatarstan, Russia), Moscow, Pero, 2016, pp. 50–59.