OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.021

ПОИСК В СТАТИЧЕСКИХ РОЯХ

Воробьев В.В.

Московский институт электроники и математики НИУ ВШЭ,
г. Москва, Россия

gatus86@mail.ru

В работе рассматривается решение проблемы организации запросов и поиска в статических роях. Данная проблема вызвана отсутствием какой-либо карты базы данных, знаний о структуре роя и т.д. Для решения данной задачи предлагается использование безадресных запросов с волновым характером их распространения по статическому рою. Это позволит существенно снизить нагрузку коммуникационных каналов и, следовательно, избежать ситуации критических загрузок в сети, образованной узлами статического роя. В основе предложенного решения лежат хорошо известные неинформированные методы поиска.
Ключевые слова: статический рой; алгоритмы поиска; запрос; неинформированный поиск.

Введение

В данный момент одним из быстроразвивающихся направлений современной робототехники является групповая робототехника, объектами изучения которой являются множества взаимодействующих между собой роботов [Карпов, 2013]. Стоит отметить, что основное отличие между этими групповыми образованиями заключается в степени информированности о других членах группы, их общих целях, задачах и т. д. [Карпов, 2011].

Данное направление в робототехнике представлено большим количеством работ, например [Konolige et al., 2006], [Dorigo et al., 2012], [McLurkin et al., 2005], [Roy et al., 2013], в которых рассмотрены различные аспекты роевого поведения, как то: движение в группе, поиск цели и т. д. Пожалуй, самыми интересными из этих работ, является [Kornienko et al., 2006], где авторы пытаются реализовать эффект эмерджентности. Данный эффект заключается в том, что система из множества простых объектов способна решать сложные задачи, которые не способен решать ни один, ни малое количество этих простых объектов.

В [Карпов, 2013] подробно рассмотрен предварительный перечень вопросов, которые необходимо решить для появления таких свойств.

Особый интерес из этого перечня вызывают вопросы организации запросов к базе данных всего роя и поиск по этой базе, которые и будут рассматриваться в этой работе. Данные задачи неразрывно связаны друг с другом, и имеет смысл рассматривать и решать их совместно.

База данных роя представляет собой накопленный опыт, наблюдаемые факты и т.п. Очевидно, что такое фрагментарное хранение отлично от того, что принято в распределенных базах данных [Карпов, 2013].

Отдельно стоит отметить тот факт, что на рой наложены ограничения, указанные в [Карпов, 2013]:

- Коммуникационные возможности членов роя серьезно ограничены, т. е. он может общаться только с фиксированным числом своих соседей.
- Статический характер роя, т. е. рассматриваются статические структуры, представляющие собой полученные в некоторый момент времени схемы соединения членов роя. Такая совокупность называется *статическим роем.*

В этом случае, ограничение по коммуникационным возможностям может серьезно повлиять на механизм формирования запроса.

Актуальность решения данной задачи заключается в том, что эффективный механизм формирования запроса и поиска по базе данных позволит подойти к решению целого ряда задач, возникающих перед статическим роем, например, к задаче выработки решения, о чем также будет сказано несколько слов.

Общие сведения о задаче и поисковых алгоритмах

Говоря об организации запросов к общей базе данных статического роя, нельзя не отметить, что путей решения данной задачи может быть два [Карпов, 2013]:

- Использование известных сетевых протоколов или их аналогов
- Отправка безадресного сообщения всем членам роя.

Однако первый путь имеет существенные недостатки, например: недостаток, связанный с физической организацией статического роя, а именно с использованием сугубо локальных связей между его членами. В таком случае посылка адресного запроса через своих соседей на периферию сопряжена с существенным возрастанием трафика. Кроме того необходима синхронизация членов роя во времени.

Другой путь заключается в отправке узлом безадресного сообщения своим соседям, которое ретранслируется каждым соседом дальше в том случае, если он не может выполнить запрос. Вычислительные эксперименты показывают, что подобная схема позволяет в асинхронном режиме гарантированно получать результаты запроса с детерминированной максимальной задержкой [Карпов, 2013].

Таким образом, для решения данной задачи лучше подходит отправка безадресного запроса всем членам роя. Однако механизм распространения запроса зависит от метода поиска в пространстве состояний. На данный момент существует множество различных методик поиска, каждая из которых имеет свои преимущества и недостатки. Поэтому имеет смысл рассмотреть данные методики и алгоритмы, которые их реализуют более подробно на предмет применимости их для решения задачи поиска в статическом рое:

- **Информированные методы** (эвристические). Характерной особенностью данных методов является то, что они используют дополнительную информацию (эвристику) о конкретной задаче, что позволяет существенно сократить перебор путем исключения заведомо бесперспективных вариантов. Информация о конкретной задаче формируется в виде *эвристической функции*. Эвристическая функция на каждом шаге перебора оценивает альтернативы на основании дополнительной информации с целью принятия решения о том, в каком направлении стоит продолжать перебор [Рассел и др., 2006].
- **Неинформированные методы** (методы слепого поиска, методы грубой силы). Данные методы не используют никаких дополнительных знаний о задаче и последовательно просматривают все состояния до тех пор, пока не будет найдено решение. Все на что способны неинформированные методы поиска, - вырабатывать преемников и отличать целевое состояние от нецелевого [Эделькамп и др., 2012].

Рассмотрим некоторые алгоритмы, относящиеся к информированным методам поиска:

- **Поиск по первому наилучшему совпадению.** Алгоритм исследует граф путем расширения наиболее перспективных узлов, выбираемых в соответствии с указанным правилом. Этим правилом является некая эвристическая оценка f(n), где n - узел графа, которая может зависеть от описания этого узла, описания цели, информации, собранной поиском на данный момент или каких-нибудь дополнительных знаний о предметной области [Pearl, 1984]. Выбирается узел с наименьшей оценкой.
- **Алгоритм A*.** Является разновидностью поиска по первому наилучшему совпадению. В данном алгоритме эвристическая оценка $f(n)=g(n)+h(n)$, где g(n) - стоимость пути от начального узла до узла n, h(n)- оценка стоимости пути от узла n до цели. Временная сложность алгоритма зависит от выбранной эвристики и в худшем случае растет экспоненциально по сравнению с длиной оптимального пути. Кроме того, требует большого объема памяти при самом неблагоприятном исходе, так как приходится запоминать экспоненциальное количество узлов.
- **Алгоритм A* с итеративным углублением** (iterative depending A*IDA*). Данный алгоритм имеет в своей основе идею итеративного углубления, применяемого в алгоритме IDDFS, который относится к неинформированным методам поиска и рассмотрен ниже. Здесь стоит отметит то, что IDA* останавливает развертывание когда оценка пути f(n), которая вычисляется также, как и в алгоритме A* превышает некую пороговую стоимость bound. Это позволяет существенно снизить потребляемые ресурсы памяти по сравнению с A*.
- **Рекурсивный поиск по первому наилучшему совпадению** (Recursive Best-First Search, RBFS). Является простым рекурсивным алгоритмом поиска, имитирующий поиск по первому наилучшему совпадению, но с использованием только линейного пространства. Алгоритм контролирует значение f(n) наилучшего альтернативного пути, доступного из любого предка текущего узла. Если значение превышается, то рекурсия отменяется и продолжает с альтернативного пути. RBFS несколько более эффективен нежели IDA*.
- **A* с ограничением памяти и упрощенный A* с ограничением памяти.** (memory-bounded A*, MA* и simplified memory-bounded SMA*). Также как IDA* и RBFS решают проблему потребления большого объема памяти, однако лишены их недостатков, например, использование не всей доступной памяти.

Информированные методы обеспечивают более эффективный поиск, но они не могут применяться для такого рода задач в статическом рое. Это

обусловлено тем, что целевой узел в данной задаче неизвестен, следовательно, включать различного рода эвристики часто не имеет смысла, так как алгоритм, реализующий такой поиск, становится неоптимальным. Кроме того, база данных всего статического роя, в общем случае, не является распределенной [Карпов, 2013], так как не имеет структуры, описывающей, в каких узлах хранятся те или иные данные, т.е. часто заранее неизвестна даже область, в которой необходимо проводить поиск. Данная особенность также влияет на целесообразность использования информированных методов поиска.

Хотя неинформированные методы поиска и решают эту задачу менее эффективно по сравнению с информированными, они являются наиболее подходящими для ее решения. Действительно, так как неизвестно в каком узле статического роя может находиться искомая продукция наиболее оправданной стратегией поиска будет полный перебор всех узлов. Рассмотрим неинформированные методы поиска подробнее:

- **Поиск в ширину** (breadth-first search, BFS). Алгоритм поиска, в котором вначале развертывается корневой узел, затем все потомки корневого узла, после этого развертываются потомки этих потомков и т. д.
- **Поиск по критерию стоимости** (uniform-cost search, UCS). В какой-то мере является расширением алгоритма поиска в ширину, который учитывает стоимость действий. Данный алгоритм будет полным и оптимальным, в случае если стоимости строго положительны.
- **Поиск в глубину** (depth-first search, DFS). В данном алгоритме поиска развертывается самый глубокий узел в текущей периферии дерева поиска, т. е. анализируется первый преемник текущего узла, затем преемник преемника и т. д.
- **Поиск с ограничением глубины** (depth-limited search, DLS). Является вариантом алгоритма поиска в глубину, где задается максимальная глубина поиска l, что позволяет решить проблему бесконечного поиска. Однако данный алгоритм мало применим, так как он не является полным, в случае если поверхностная цель d лежит за границей поиска l d>l и не оптимален в случае d<l. Является основой поиска в глубину с итеративным углублением.
- **Поиск в глубину с итеративным углублением** (iterative-deepening depth-first search, IDDFS, DFID). В своей основе имеет алгоритм DLS с возможностью увеличения границы поиска l в случае если цель не была найдена. Сочетает в себе преимущества DFS (пространственная сложность O(b*l)) и BFS (полнота и оптимальность при конечном b).

Однако не стоит забывать о принципиальной возможности использования информированных методов поиска после нескольких итераций поиска с помощью методов неинформированных. Таким образом, поисковый запрос в статическом рое должен формироваться на основе неинформированных методах поиска причем запрос должен распространяться на безадресной основе.

Формирование запроса и поиск в статическом рое

Рассмотрим следующую задачу: существует N=m*n членов статического роя, связанных друг с другом локальными связями (рис. 1).

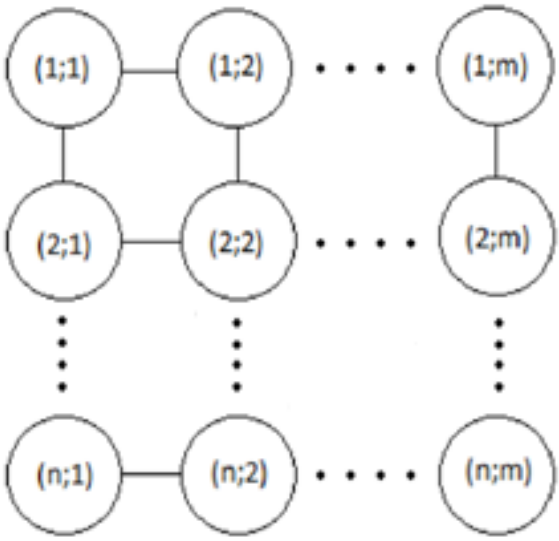

Рисунок 1 – Физическая структура статического роя

Стоит отметить тот факт, что с точки зрения каждого члена роя в его составе находится только он сам и его соседи. Это является следствием ограничения на взаимодействие друг с другом.

Существует инициирующий узел, который, например, был выбран с помощью процедуры, описанной в [Карпов, 2014]. Данный узел либо сам формирует запросы, либо имеет возможность получать их извне. С точки зрения организации процедуры выполнения запроса не имеет значения, откуда он появился. Важно лишь то, что его формат поддерживается данной системой и то, что после получения/формирования запроса, инициирующий узел не может получать/формировать новый до тех пор, пока запрос не будет выполнен или станет понятно, что он невыполним.

Так как уже было сказано, что запрос является безадресным, и на начальном этапе не имеет смысла использовать информированные методы поиска, то, инициирующий узел, в случае неудачной попытки его обработать (ничего не найдено в его БД), передаст его **всем** своим соседям. Соседи также попытаются выполнить запрос, а в случае неудачи, отправят его своим соседям и т.д.. Такой подход наиболее похож на алгоритм **поиска в глубину с итеративным углублением** (IDDFS). Отличие заключается в том, что благодаря тому, что каждый сосед инициирующего узла это отдельный вычислительный блок, мы можем отправить запрос одновременно всем соседям, которые, при необходимости, увеличат глубину поиска.

Однако может появиться такая ситуация, в которой узел попытается передать запрос узлу, который этот запрос уже обработал. Такая коллизия решается путем введения трех правил:

- Отправляющий узел знает количество своих потомков, которым был отправлен запрос, и запоминает его.
- Узел не отправляет запрос узлу, от которого он его получил. То есть реализуется функция, позволяющая запомнить, откуда пришел запрос.
- В случае если узел обработал запрос и, по прошествии некоторого времени, снова получает тот же запрос, он отвечает на него специальным сообщением. Отправляющий узел, приняв это сообщение, уменьшает число своих потомков на единицу.

Таким образом, запрос постепенно уйдет от инициирующего узла на периферию роя, сформировав дерево поиска и не загружая при этом каналы связи, так как назад запрос уйти не может из-за правила №2, а соседям, которые его уже обработали из-за правила №3 пройдет сначала запрос, а потом ответ на него (рис. 2). Инициирующий узел (ИУ) посылает запрос (одинарная стрелка) своим соседям, те посылают его своим соседям и т.д. Двойными стрелками показана отправка запроса узлу, который его уже принял.

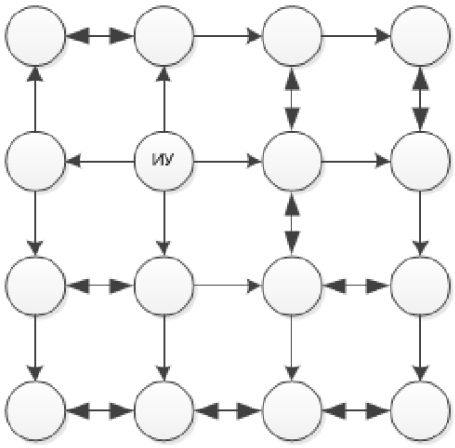

Рисунок 2 – Отправка запроса

В случае если какой-либо узел смог выполнить запрос, он формирует сообщение, механизм распространения которого полностью аналогичен механизму распространения поискового запроса. Это сообщение несет в себе метку о том, что поиск закончен и результат этого поиска. Метка блокирует узлы, которые получили это сообщение для получения поискового запроса, который был выполнен и для получения другого ответа на этот запрос. Когда это сообщение доходит до инициирующего узла, он способен снова получать/формировать новый запрос.

Для того, чтобы избежать ситуации с “зависанием поиска”, которая характерна тем, что запрос не может быть выполнен (например, в базе данных статического роя нет такой информации) реализуется следующий механизм: в случае если периферийный узел не имеет потомков (исходя из вышеописанных правил), то он отправляет соответствующее сообщение узлу-предку. Как только предок получил такие сообщение от всех своих потомков, он передает такое же сообщение своему предку и т. д. Как только инициирующий узел получил такие сообщения от всех своих потомков, он может принимать/формировать другой запрос.

Однако механизм формирования, распространения и выполнения запроса в статическом рое несет сугубо вспомогательную функцию по отношению к другим задачам, например, задаче выработки общего решения. Полагается, что правило является множеством импликаций, вида $c_1 \& c_2 \& \ldots \& c_n \to f$. Инициирующий узел будет пытаться отработать такое правило и поочередно отправлять запросы в сеть на определение истинности каждого из конъюнктов. Узлы будут искать в своих БД интересующие инициатора данные, которые могут быть либо фактами, либо правыми частями неких хранящихся в узлах правил. Реализация такого механизма в духе ПРОЛОГа, не смотря на свою универсальность, ведет к тому, что все ресурсы сети, и особенно коммуникационных каналов, будут заняты исключительно логическим выводом [Карпов, 2013].

Однако использование такого рода запроса, позволяет отчасти решить проблему сильной загрузки канала связи. Кроме того, с таким механизмом формирования запроса достаточно легко реализуется функция управления поиском в ПРОЛОГе с использованием предикатов fail и cut.

Предикат fail реализуется путем добавления к запросу соответствующего аргумента, который позволяет выполнившему запрос узлу не формировать блокирующее сообщение и передать запрос далее.

Механизм работы предиката cut в такой системе полностью аналогичен – в аргументе запроса указан соответствующий параметр, при нахождении которого дальнейший поиск блокируется. Также имеется возможность удалить другие результаты поиска, которые были найдены до того как сработал cut.

Кроме того, если запрос выполнен, но ответ также является конъюнкцией продукций, то инициирующий узел передает управление узлу, содержащему ответ, который в свою очередь также отправляет свой запрос на поиск соответствующих конъюнкций, то есть осуществляется рекурсия.

Характеристики поискового алгоритма при его использовании в статическом рое

Рассмотрим ряд характеристик работы данного алгоритма в условиях статического роя:

- **Полнота**. Так как данный алгоритм является алгоритмом IDDFS, который применяется в условиях статического роя, то он является полным [Рассел и др., 2006].
- **Оптимальность**. Данный алгоритм оптимален, и эта характеристика также вытекает из оптимальности IDDFS [Рассел и др., 2006]. Так как в статическом рое количество коммуникационных каналов его членов ограничено, то он оптимален всегда.
- **Пространственная сложность**. Очевидно, что пространственная сложность данного алгоритма будет не более, чем у IDDFS и будет равна O(b*l), где b – коэффициент ветвления, а l – глубина поиска [Рассел и др., 2006]. При этом b ограничено количеством коммуникационных каналов каждого члена роя.
- **Временная сложность**. Временная сложность в условиях применения в статическом рое будет зависеть исключительно от глубины поиска l,коэффициента связи k и средней величины базы данных узла N, и будет равна $O(l*k \oplus N)$

Коэффициент связи необходим для более точного определения временной сложности, так как время поиска в статическом рое сильно зависит от времени распространения запроса, то есть, от качества каналов связи.

Рассмотрим процедуру поиска на примере. В момент времени T=1 запрос приходит к инициирующему узлу (рис. 3). Узел сразу отправляет запрос своим узлам-потомкам и начинает обрабатывать запрос. Узлы-потомки, в свою очередь, производят те же действия.

```
---------------Level#0----------------
#=1 (3 5 3 2 9 6 2 7 4 0 )Status: received a request
Clock #=1
```

Рисунок 3 – Начало обработки запроса

На рисунке 4 видно, что запрос дошел до периферийных узлов, и искомая информация была найдена.

```
---------------Level#0----------------
#=1 (3 5 3 2 9 6 2 7 4 0 |0| ))Status: request not found
---------------Level#1----------------
#=2 (6 1 9 8 0 4 9 3 0 6 |6| ))Status: request not found
#=3 (4 9 5 2 7 4 7 0 4 2 |2| ))Status: request not found
---------------Level#2----------------
#=4 (0 2 6 7 9 7 8 1 1 8 |8| ))Status: request not found
#=5 (8 0 8 0 4 6 1 0 4 7 |7| ))Status: request not found
#=6 (7 7 6 8 9 6 2 9 8 1 |1| ))Status: request not found
#=7 (5 5 4 7 8 2 7 0 0 3 |3| ))Status: request not found
---------------Level#3----------------
#=8 (4 0 4 3 8 2 6 3 0 |0| ))Status: request not found
#=9 (9 4 3 2 6 8 0 6 8 |8| ))Status: request not found
#=10 (5 7 9 9 3 1 1 5 7 |7| ))Status: request not found
#=11 (1 6 3 4 8 1 5 3 1 |1| ))Status: request not found
#=12 (7 3 1 5 6 0 3 4 4 |4| ))Status: request not found
#=13 (4 3 4 0 3 4 0 9 7 |7| ))Status: request not found
#=14 (6 2 6 0 2 2 1 3 5 |5| ))Status: request not found
#=15 (3 2 2 3 6 1 7 2 11 )Status: request is done. Fact is on 3 level
Clock #=7
```

Рисунок 4 – Искомая информация найдена

На рисунке 5 видно как запрос дошел до инициирующего узла.

```
---------------Level#0----------------Fact is on this level
#=1 (3 5 3 2 9 6 2 7 4 0 )Status: request not found
---------------Level#1----------------
#=2 (6 1 9 8 0 4 9 3 0 6 )Status: request not found
#=3 (4 9 5 2 7 4 7 0 4 2 )Status: request not found
---------------Level#2----------------
#=4 (0 2 6 7 9 7 8 1 1 8 )Status: request not found
#=5 (8 0 8 0 4 6 1 0 4 7 )Status: request not found
#=6 (7 7 6 8 9 6 2 9 8 1 )Status: request not found
#=7 (5 5 4 7 8 2 7 0 0 3 )Status: request not found
---------------Level#3----------------
#=8 (2 4 0 4 3 8 2 6 3 0 )Status: request not found
#=9 (7 9 4 3 2 6 8 0 6 8 )Status: request not found
#=10 (5 5 7 9 9 3 1 1 5 7 )Status: request not found
#=11 (4 1 6 3 4 8 1 5 3 1 )Status: request not found
#=12 (2 7 3 1 5 6 0 3 4 4 )Status: request not found
#=13 (1 4 3 4 0 3 4 0 9 7 )Status: request not found
#=14 (9 6 2 6 0 2 2 1 3 5 )Status: request not found
#=15 (9 3 2 2 3 6 1 7 2 11 )Status: request is done
Clock #=10
```

Рисунок 5. Запрос выполнен

Заключение

Таким образом, предложенная методика реализации запросов внутри статического роя позволяет решить следующие задачи:

- Производить поиск необходимых элементов внутри общей базы данных статического роя.
- Находить все возможные альтернативы искомого элемента.
- Производить рекурсивный поиск в случае, если искомый элемент оказался не фактом, а, в свою очередь, также множеством импликаций.

Методика имеет в своей основе хорошо известный алгоритм неинформированного поиска IDDFS и формирует запросы на безадресной основе, что позволяет не загружать слабые коммуникационные каналы. Кроме того, такой механизм формирования запросов позволяет вообще не загружать эти каналы, даже в случае рекурсивного поиска. Однако это влечет за собой то, что при достаточно большом числе узлов статического роя или их неудачной конфигурации (например, узлы расположены в линию и инициирующий узел находится в начале этой линии) может возникать ситуация с их простоем, т.е., обработав и передав запрос дальше, узлы будут ждать завершения его выполнения. Данная проблема будет рассматриваться в дальнейшем.

Еще одним немаловажным аспектом работы является то, что такой механизм может позволить подойти к решению проблемы выработки общего решения с использованием ПРОЛОГ. Как уже было упомянуто, одним из препятствий к его использованию является слабость коммуникационных каналов статического роя. Такой механизм формирования запросов может оказаться решением данной проблемы.

Библиографический список:

[Карпов,2011] Карпов В.Э. Коллективное поведение роботов. Желаемое и действительное //Современная мехатроника. Сб. научн. трудов Всероссийской научной школы (г.Орехово-Зуево, 22-23 сентября 2011) -Орехово-Зуево,2011. -С132 с. сс.35-51.

[Карпов, 2013] 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) Сб. научных трудов. В 3-т., М: Физматлит, 2013, Т.2, с.730-739

[Карпов, 2014] Карпов В.Э. Процедура голосования в однородных коллективах роботов //XIV национальная конференция по искусственному интеллекту с международным участием КИИ-2014 (24-27 октября 2014 г., Казань, Россия): Труды конференции. В 3-т., Казань: Изд-во РИЦ «Школа», 2014, Т.2, с.159-167, -341с.

[Рассел и др., 2006] Стюарт Рассел, Питер Норвиг Искусственный интеллект: современный подход // Artificial Intelligence: A Modern Approach.- 2-е изд. - М: Вильямс, 2006. - 1408 с. - ISBN 5-8459-0887-6.

[Dorigo et al., 2012] M. Dorigo, D. Floreano,L.M. Gambardella,F. Mondada et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms // IEEE Robotics & Automation Magazine, Pages In press, 2012

[Konolige et al., 2006] K. Konolige, D. Fox, C. Ortiz, et al. Centibots: Very large scale distributed robotic teams // Springer Tracts in Advanced Robotics Volume 21, 2006, pp 131-140

[Kornienko et al., 2006] S. Kornienko, O. Kornienko, P. Levi Swarm embodiment - a new way for deriving emergent behavior in artificial swarms //Autonome Mobile Systeme 2005 (AMS05), 25-32, DOI: 10.1007/3-540-30292-1_4, 2006

[McLurkin et al., 2005] J. McLurkin, D. Yamins Dynamic Task Assignment in Robot Swarms // Robotics: Science and Systems Conference, June 8, 2005

[Pearl, 1984] Pearl J. Heuristics: Intelligent Search Strategies for Computer Problem Solving. - Addison-Wesley, 1984. - . 48.

[Roy et al., 2013] N. Roy, P. Newman, S. Srinivasa Towards A Swarm of Agile Micro Quadrotors.//Robotics:Science and Systems VIII - MIT Press 1, 2013

THE SEARCHING TASK IN THE STATIC SWARM

Vorobiev V.V.

National Research University The Higher School of Economics,Moscow, Russian Federation

gatus86@mail.ru

This paper considers the problem of organization and search queries in static swarms. This problem is caused by the absence of any map database of knowledge about the structure of the swarm, etc. To solve this problem we suggest the use of non-address requests to the wave nature of the distribution of the static swarm. This will significantly reduce the load of communication channels, and hence avoid the situation critical loads the network formed by nodes of the static swarm. At the heart of the proposed solutions are well known uninformed search methods

.