УДК 007.52:611.81

В.Ф. Гузик, А.А. Приемко

МИКРОПРОЦЕССОРНАЯ РЕАЛИЗАЦИЯ НЕЙРОСЕТЕВОГО ПЛАНИРОВЩИКА ТРАЕКТОРИИ ДВИЖЕНИЯ АДАПТИВНОГО МОБИЛЬНОГО РОБОТА С ИСПОЛЬЗОВАНИЕМ КАРТЫ ВНЕШНЕЙ СРЕДЫ

Рассматривается решение задачи планирования траектории движения с использованием карты внешней среды для интеллектуальной системы управления адаптивного мобильного робота. Трудность при создании таких систем состоит в необходимости постоянного анализа больших объемов информации о состоянии внешней среды и определения направления движения к цели. Среда, в общем случае, является динамически изменяющейся и плохо формализованной. Поэтому традиционные подходы, основанные на применении последовательных алгоритмов обработки данных, оказались неэффективными, что делает актуальной тему разработки бионических методов управления. В статье описывается алгоритм работы бионической системы управления. Эта система решает задачу движения по карте среды и ориентирована на использование аппаратно реализованных нейропроцессорных управляющих сетей. Основной проблемой, возникающей при использовании описанной системы, является необходимость масштабирования карты перед ее отображением в нейропроцессорную сеть, что существенно влияет на оптимальность вырабатываемой траектории. Решение данной проблемы в рамках аппаратной реализации затруднительно. Новизна работы состоит в разработке алгоритма, позволяющего избежать масштабирования, основанного на программной эмуляции работы нейросети определения направления движения. Данный алгоритм ориентирован на параллельную обработку данных о внешней среде. Приводятся результаты исследования алгоритма на многоядерной вычислительной системе.

Планирование траектории; нейросетевая система управления; параллельная обработка данных.

V.F. Guzik, A.A. Priemko

MICROPROCESSOR REALIZATION OF THE NEURONETWORK TRAJECTORY PLANNER FOR THE ADAPTIVE MOBILE ROBOT WITH USING OF THE ENVIRONMENT MAP

The article is devoted to solving the problem of path planning with using of environment map for intellectual control system of adaptive mobile robot. Difficulty at creation of such systems consists in need of the continuous analysis of large volumes of information about a state of external environment and planning a trajectory of movement in it. Environment, generally, is dynamically changing and badly formalized. Therefore the traditional approaches based on application of consecutive algorithms of data processing are inefficient. This circumstance makes actual the subject of development of bionic control methods. In the article the algorithm of a bionic control system is described. This system solves a problem of movement direction determination basing on map of the environment. This system is focused on use of hardware implemented neuroprocessor control networks. The main problem arising at use of system is the necessity of map scaling before displaying to a neuroprocessor network. This greatly influences to an optimality of a planned trajectory. The solution of this problem within hardware realization is difficult. Novelty of work consists in development of the algorithm, allowing to avoid the scaling, based on program emulation of work of a neuronet of definition of the direction of movement. This algorithm is focused on parallel data processing of the environment map. Results of investigation of algorithm on the multicore computer system are given.

The path planning; the neuronet control system; the parallel data processing.

Введение. В настоящее время интенсивно развивается область исследований, связанная с разработкой адаптивных мобильных роботов (АМР) с элементами искусственного интеллекта. Такие АМР должны функционировать в динамической априори неформализованной внешней среде. Поэтому их система управления должна решать навигационные задачи, в том числе определять направление движения к цели, отсутствующей в поле зрения его сенсорной подсистемы (СП) с использованием карты внешней среды. Основная трудность при создании интеллектуальных подсистем планирования состоит в необходимости постоянного анализа больших объемов информации о состоянии внешней среды и выработке траектории перемещения в ней. Среда, в общем случае, является динамически изменяющейся и плохо формализованной. Поэтому традиционные подходы, основанные на применении последовательных алгоритмов обработки данных, оказались неэффективными. В то же время известно, что нервная система живых организмов хорошо справляется с данной задачей, что послужило причиной разработки бионических методов управления.

Суть бионического метода к созданию искусственного интеллекта АМР состоит в технической имитации биологических систем, обеспечивающих разумное поведение в сложной, априори, неформализованной внешней среде [1]. Данный подход базируется на выявлении и использовании биологических аналогий, в частности на использовании результатов нейрофизиологических и нейрокибернетических экспериментов по изучению нервной системы человека и животных.

В соответствии с бионическим методом в качестве теоретической модели процессов адаптивного управления используются данные нейрофизиологии о функционировании нервной системы и прежде всего данные о функциональной системе П.К. Анохина [1]. Согласно теории функциональной системы, механизмы, обеспечивающие целенаправленное адаптивное поведение организма, имеют определенную структуру, включающую последовательность процессов, в том числе афферентный синтез, принятие решения, формирование программы действий и ее выполнение. Афферентный синтез включает в себя обработку информации о состоянии внешней среды, о положении в ней организма и его состоянии, а также о целях, которые преследует организм в текущей ситуации. Принятие решения заключается в выборе действия, направленного на достижение цели в сложившихся условиях.

Один из алгоритмов решения задачи планирования траектории с использованием бионического подхода описан в работе [1]. Экспериментальные исследования данного алгоритма [2, 3] показали его эффективность. Суть алгоритма состоит в том, что определение направления движения на каждом шаге производится в 2 этапа. На первом этапе карта среды из долговременной памяти отображается в масштабирующую структуру с учетом текущего положения робота на ней, получаемого от подсистемы определения собственного положения. Так как мощность множества карты в общем случае значительно превышает мощность множества процессоров сети афферентного синтеза (САС), то перед отображением карты в САС ее необходимо масштабировать. Далее САС определяет направление перемещения к цели по карте при помощи волнового алгоритма [1] без его отработки эффекторной подсистемой. Информация об этом направлении с выходов сети принятия решения (СПР) отображается в тот процессор САС, который соответствует выбранному направлению. На втором этапе в САС отображается информация из сенсорной подсистемы, формализованная в виде плана проходимости подсистемой формирования модели внешней среды. Решение принятое СПР на втором этапе отрабатывается эффекторной подсистемой, которая перемещает робот на один шаг в направлении к цели. Аппаратно реализованная САС для АМР, функционирую-

щего в двумерном пространстве, представляет собой матрицу нейроэлементов (НЭ), соединенных по четырехточечному шаблону соседства. Каждый НЭ отображает информацию о проходимости соответствующего участка внешней среды и работает следующим образом [1]. Если НЭ отображает положение цели, то он генерирует сигнал возбуждения. Если НЭ отображает положение препятствия, то он блокирует прохождение сигнала возбуждения. Если НЭ отображает положение свободного участка внешней среды, то он пропускает сигналы возбуждения от соседних НЭ с задержкой, пропорциональной трудности прохождения соответствующего участка внешней среды.

Основным недостатком описанного алгоритма является необходимость масштабирования карты перед ее отображением в САС, что неизбежно ведет к искажению информации о положении препятствий и цели и, как следствие, к неоптимальному выбору направления движения к цели на первом этапе рассмотренного выше алгоритма. Данный недостаток затруднительно преодолеть в рамках аппаратной реализации САС, если множество элементов карты значительно превосходит множество ее процессоров. Однако из-за постоянного роста вычислительных мощностей ЭВМ общего назначения они все чаще используются для программной реализации алгоритмов управления АМР. Поэтому в данной статье рассматривается модификация рассмотренного выше алгоритма без использования операции масштабирования карты. Суть предлагаемого алгоритма состоит в программной эмуляции САС, размерность которой равна размерности карты внешней среды, ориентированной на параллельную эмуляцию работу НЭ на современных многоядерных системах. При этом отпадает необходимость в масштабировании карты среды.

Пусть позиция каждого НЭ в САС задается в декартовой системе координат, как показано на рис. 1.

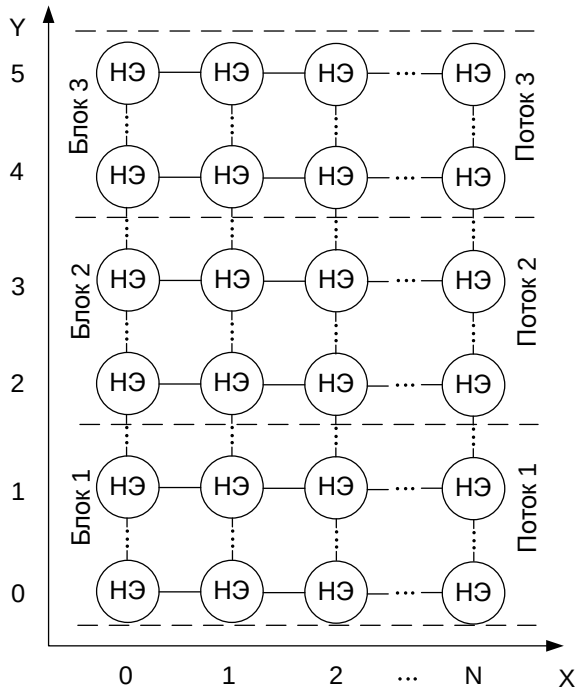

Рис. 1. Распределение нагрузки при обработке карты несколькими потоками

Тогда разделим множество элементов карты построчно на L блоков так, чтобы в каждом блоке было примерно одинаковое количество строк. Среднее количество строк в наборе определяется как частное от деления общего количества строк в карте ($Y_{\max}$) на количество процессорных ядер ($NumThreads$), обрабатывающих наборы. Предполагается, что каждый набор обрабатывается отдельным ядром. При этом каждый программный поток, выполняемый ядром, обрабатывает непрерывную последовательность строк в диапазоне $[LowBound, HighBound]$. Пусть карта проходимости среды хранится в долговременной памяти и представляет собой массив $ObstacleArray$, состоящий из $Y_{\max}$ строк и $X_{\max}$ столбцов. Значение каждого элемента хранит информацию о проходимости соответствующего участка среды в следующем виде: $ObstacleArray[X][Y] = 0$, если соответствующий участок внешней среды содержит препятствие, $ObstacleArray[X][Y] = [1..K]$, если участок свободен для прохождения.

При этом хранимое значение элемента прямо пропорционально трудности проходимости соответствующего участка внешней среды. Кроме того, введем дополнительный массив $T\arg etArray$, размерность которого совпадает с размерностью массива карты. Каждый элемент $T\arg etArray[X][Y]$ принимает единичное значение, если через соответствующий элемент карты проходила волна возбуждения, и нулевое в противном случае. Тогда алгоритм определения направления движения к цели по карте имеет следующий вид.

1. Получить координаты множества участков $CurrentStep$ волнового фронта на текущем шаге.

2. Если множество $CurrentStep$ не пусто, то переход к п.3, иначе к п.19.

3. Получить координаты очередного элемента (X,Y) из $CurrentStep$.

4. Если $ObstacleArray[X][Y] > 1$, то

$ObstacleArray[X][Y] = ObstacleArray[X][Y] - 1$; перейти к п.3, иначе к п.5.

5. Если $ObstacleArray[X][Y] \neq 0 \,\&\, T\arg etArray[X][Y] = 0$, то перейти к п.6, иначе к п.18.

6. Если $X - 1 \geq 0 \,\&\, ObstacleArray[X-1][Y] \neq 0 \,\&$

$T\arg etArray[X-1][Y] = 0$, то добавить элемент $(X-1, Y)$ в множество участков волнового фронта на следующем шаге $NextStep$.

7. Если $X + 1 < X_{\max} \,\&\, ObstacleArray[X+1][Y] \neq 0 \,\&$

$T\arg etArray[X+1][Y] = 0$, то добавить элемент $(X+1, Y)$ в $NextStep$.

8. Если $Y - 1 \geq 0 \,\&\, ObstacleArray[X][Y-1] \neq 0 \,\&$

$T\arg etArray[X][Y-1] = 0$, то перейти к п.9, иначе к п.12.

9. Если $Y - 1 < LowBound$, то определить номер потока ($nThread$), обрабатывающего строку $Y - 1$ карты.

10. Если поток с номером $nThread$ не активен, то активировать его, передав $(X, Y-1)$ в качестве начальной точки распространения волны, перейти к п.12, иначе к п.11.

11. Добавить элемент $(X, Y-1)$ в $NextStep$ для текущего потока.

12. Если $Y-1<0$, то сохранить значение X, конец; иначе переход к п.13;

13. Если $Y+1<Y_{\max}$ & $ObstacleArray[X][Y+1]\neq 0$ & $Target Array[X][Y+1]=0$, то перейти к п.14, иначе к п.17.

14. Если $Y+1>HighBound$, то определить номер потока $nThread$, обрабатывающего строку $Y+1$ карты.

15. Если поток с номером $nThread$ не активен, то активировать его, передав $(X,Y+1)$ в качестве начальной точки распространения волны, перейти к п.17, иначе к п.16.

16. Добавить элемент $(X,Y+1)$ в $NextStep$ для текущего потока.

17. Установить $Target Array[X][Y]$ в 1.

18. Удалить элемент (X,Y) из $CurrentStep$, перейти к п.2.

19. $CurrentStep = NextStep$.

20. Если множество $CurrentStep$ не пусто, переход к п.1, иначе к п.21.

21. Конец.

Суть данного алгоритма состоит в том, что координаты целевого участка заносятся в множество $CurrentStep$, хранящее участки фронта волны возбуждения на текущем шаге распространения волны. Перед началом работы алгоритма в $CurrentStep$ заносится позиция целевого элемента карты. После этого в пп. 3 и 4 для каждого участка из $CurrentStep$ происходит проверка значения трудности его прохождения. Если это значение больше единичного, то происходит его уменьшение на единицу и переход к проверке следующего элемента $CurrentStep$. Тем самым моделируется задержка прохождения сигнала через НЭ, пропорциональная трудности преодоления соответствующего участка среды. В противном случае в п. 5 осуществляется генерация координат участков волнового фронта на следующем шаге. Данное действие осуществляется в соответствии с четырехточечным шаблоном соседства. При этом элемент карты попадает в волновой фронт на следующем шаге ($NextStep$) в том случае, если соответствующий ему участок внешней среды не содержит препятствие ($ObstacleArray[X][Y]\neq 0$) и он еще не обработан ($Target Array[X][Y]=0$). Кроме того, при распространении волны в горизонтальном направлении, происходит проверка ее выхода за пределы карты ($X-1\geq 0$ и $X+1<X_{\max}$) в пп. 6 и 7. Распространение волны в вертикальном направлении происходит в пунктах 8–15. При этом в п. 9 вначале проверяется, не вышла ли волна за нижнюю границу ($LowBound$) блока, обрабатываемого текущим программным потоком. В случае выхода определяется номер целевого потока ($nThread$). Если он не активен (п. 10), то происходит его активация с передачей параметров соответствующего участка волнового фронта. В противном случае участок волнового фронта обрабатывается в текущем потоке для того, чтобы избежать временных расходов, связанных с синхронизацией потоков (п. 11). Если волна дошла до элемента, отображающего собственное положение АМР (п. 12), то происходит сохранение координата X участка, через который пришла волна, и завершение работы алгоритма. Аналогичные действия происходят, когда волна выходит за верхнюю границу ($HighBound$) блока (пп. 14–16). После этого в пункте 17 происходит фиксация факта прохождения волны через рассмотренный элемент карты, затем происходит ее удаление из множества $CurrentStep$. После этого

описанные выше действия повторяются для всех элементов *CurrentStep*. После отработки всех элементов текущего множества в него копируются элементы фронта волны на следующем шаге, и описанные действия повторяются. Работа алгоритма завершается, если волна возбуждения дошла до элемента, отображающего собственное положение АМР либо прошла все незанятые участки в рамках своего блока. Основным достоинством описанного алгоритма является его масштабируемость под произвольное количество процессорных ядер.

Исследования приведенного алгоритма проходили на программной модели, написанной на языке С++. В качестве аппаратной платформы использовался ПК на базе шестиядерного процессора AMD Phenom2 X6 с 4ГБ ОЗУ под управлением OCWindows 7. Для исследований использовались карты с количеством участков 10^4, 2×10^4, 6×10^4, 12×10^4 и 50×10^4. Для каждой карты проводилось измерение времени распространения волны с использованием от одного до шести процессорных ядер. Результаты моделирования (в миллисекундах) приведены в табл. 1.

Таблица 1

Результаты экспериментальных исследований алгоритма

Кол-во элементов карты	10^4	2×10^4	6×10^4	12×10^4	50×10^4
1 поток	20	38	85	155	680
2 потока	11	19	46	78	346
3 потока	7	13	32	53	235
4 потока	6	11	21	41	169
5 потоков	4	7	17	34	139
6 потоков	3	6	15	23	121

Выводы. Как видно из результатов моделирования, наблюдается практически линейное уменьшение временных затрат при увеличении количества процессорных ядер, используемых для определения направления движения. Это позволяет сделать вывод о целесообразности применения описанного алгоритма в системах управления АМР.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Чернухин Ю.В.* Нейропроцессорные сети: Монография. – Таганрог, 1999. – 439 с.
2. *Чернухин Ю.В., Приемко А.А.* Моделирование поведения интеллектуальных агентов в динамических средах: Учебное пособие. – Таганрог: ТТИ ЮФУ, 2007. – 244 с.
3. *Чернухин Ю. В., Приемко А.А., Сапрыкин Р.В.* Программно-аппаратное моделирование интеллектуального поведения мобильных роботов на базе робототехнического комплекта “Hemisson”: Учебное пособие. – Таганрог. ТТИ ЮФУ, 2010. – 335 с.

Статью рекомендовал к опубликованию д.т.н., профессор С.В. Тарарыкин.

Гузик Вячеслав Филиппович – Южный федеральный университет; e-mail: gvf@tsure.ru; 347928, г. Таганрог, пер. Некрасовский, 44; тел.: 88634371656; кафедра вычислительной техники; зав. кафедрой; д.т.н.; профессор.

Приемко Андрей Анатольевич – e-mail: andrewprmko@mail.ru; кафедра вычислительной техники; к.т.н.; доцент.

Guzik Vyatcheslav Filippovich – Southern Federal University; e-mail: gvf@tsure.ru; 44, Nekrasovskiy, Taganrog, 347928, Russia; phone: +78634371656; the department of computer engineering; head of department; dr. of eng. sc.; professor.

Priemko Andrey Anatolievich – e-mail: andrewprmko@mail.ru; the department of computer engineering; associate professor.