Гибридный алгоритм распараллеливания для численного решения задачи синтеза управления

А.И. Дивеев[2], А.С. Уваров[1], Д.Б. Хамадияров[1]

ФГБОУ Российский университет дружбы народов[1], Федеральное государственное бюджетное учреждение науки Вычислительный центр им. А.А. Дородницына Российской академии наук[2]

В статье рассматриваются алгоритмы распараллеливания численного метода синтеза робастных систем управления. Метод синтеза заключается в замене множества неопределенностей конечным множеством точек из области неопределенности. Расчет функционалов производится для каждой точки из этого множества. Чем больше точек в множестве неопределенностей, тем выше качество синтезируемой системы. Время вычислений увеличивается пропорционально количеству точек. С целью повышения эффективности расчета используются гибридные методы параллельных вычислений. Проведен анализ эффективности алгоритма при разных параметрах расчета на примере задачи синтеза робастного управления спуском космического аппарата на Луну в условиях параметрической неопределенности.

1. Введение

Работа посвящена ускорению и увеличению эффективности алгоритма численному синтеза системы управления. Синтез системы управления обеспечивает нахождение функции, описывающей зависимость управления от координат пространства состояний объекта. Рассматривается задача синтеза робастной системы управления, в которой модель объекта управления имеет неопределенные параметры. Для решения задачи применяется метод сетевого оператора [1, 2, 6], который использует кодирование математического выражения в форме целочисленной матрицы. Метод сетевого оператора относится к классу современных методов символьной регрессии. Такие методы производят поиск оптимального решения в форме кода с помощью эволюционных алгоритмов [4, 7]. Эффективность этих алгоритмов определяется мощностью начального множества возможных решений и количеством циклов преобразования этого множества. С целью обеспечения свойства робастности каждый неопределенный параметр задается конечным множеством точек. Подобное множество неопределенных значений существенно усложняет и замедляет процесс поиска решения. Для повышения эффективности метода сетевого оператора применяются гибридные подходы распараллеливания. В работе представлен анализ параллельных алгоритмов метода сетевого оператора. Рассматриваются подходы, обеспечивающие параллельное вычисление функционалов на множестве значений неопределенных параметров и параллельное вычисление вариаций для множества базисных решений [3, 8 - 10]. Рассматривается пример синтеза робастной системы управления спуском космического аппарата (КА) на поверхность Луны.

2. Задача численного синтеза робастного управления

Модель объекта управления задана системой обыкновенных дифференциальных уравнений:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{u}), \qquad (1)$$

где функция $\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{u})$ - вектор-функция с векторными аргументами, $\mathbf{x}$ – вектор состояния, $\mathbf{x} \in \mathrm{R}^n$, $\mathbf{u}$ – вектор управления, $\mathbf{u} \in \mathrm{U} \subseteq \mathrm{R}^m$, $\mathrm{n} \geq \mathrm{m}$, $\mathbf{y}$ – вектор неопределенных параметров, $\mathbf{y} \in \mathrm{Y} \subseteq \mathrm{R}^l$, U, Y - замкнутые и ограниченные множества.

Для системы (1) заданы начальные условия, значения которых могут зависеть от неопределенных параметров:

$$\mathbf{x}(0) = \mathbf{g}^0(\mathbf{x}^0, \mathbf{y}), \tag{2}$$

где $\mathbf{x}^0$ – вектор заданных начальных значений.

Заданы терминальные условия:

$$\varphi_i(\mathbf{x}(t_f)) = 0, i = \overline{1,k}, k \le n, \tag{3}$$

где t_f – время процесса управления определяется из соотношения:

$$t_f = \begin{cases} t, \max\limits_i \left\{ \left| \varphi_i(\mathbf{x}(t_f)) \right| : i = \overline{1,k} \right\} = 0, t \le t^+ \\ t^+, \text{иначе} \end{cases}, \tag{4}$$

где t^+ задано и определяет верхнюю границу возможного времени управления.

Задан функционал качества:

$$J = \int_0^{t_f} f_0(\mathbf{x}(t), \mathbf{u}(t))dt \to \min, \tag{5}$$

Необходимо найти функцию управления в виде:

$$\mathbf{u} = \mathbf{h}(\mathbf{x}), \tag{6}$$

где $\mathbf{h}(\mathbf{x})$ - вектор-функция $\mathbf{h}(\mathbf{x}) : \mathrm{R}^n \to \mathrm{R}^m$ со следующими свойствами:

$$\mathbf{h}(\mathbf{x}(t, \mathbf{y})) \in \mathrm{U},$$

$$\int_0^{t_f} f_0(\mathbf{x}(t, \mathbf{y}), \mathbf{h}(\mathbf{x}(t, \mathbf{y})))dt = \min_{\mathbf{u} \in \mathrm{U}} \int_0^{t_f} f_0(\mathbf{x}(t, \mathbf{y}), \mathbf{u})dt, \tag{7}$$

$$\varphi_i(\mathbf{x}(t_f, \mathbf{y})) = 0, i = \overline{1,k},$$

где $\mathbf{x}(t, \mathbf{y})$ - решение системы дифференциальных уравнений $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{h}(\mathbf{x}))$, $\forall \mathbf{y} \in \mathrm{Y}$.

Для решения данной задачи (1-6) используем численный метод сетевого оператора. Множество неопределенных параметров заменяем конечным множеством определенных значений из заданного множества:

$$\widetilde{\mathrm{Y}} = \left\{ \mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P \right\}. \tag{8}$$

Заменяем терминальные условия (3) дополнительным функционалом, а функционал качества (5) заменяем суммой функционалов, вычисленных для каждой точки из множества неопределенных параметров (8). В результате получаем задачу многокритериальной оптимизации с критериями:

$$J_1 = \sum_{i=1}^{P} \left(\sum_{j=1}^{k} \int_0^{t_f} f_0(\mathbf{x}(t), \mathbf{u}(t))dt \right)_{y^i} \to \min, \tag{9}$$

$$J_2 = \sum_{i=1}^{P} \left(\sum_{j=1}^{k} \left| \varphi_j(\mathbf{x}(t)) \right| \right)_{y^i} \to \min. \tag{10}$$

3. Метод сетевого оператора

Метод сетевого оператора был описан во многих работах [1, 2, 6]. Метод сетевого оператора позволяет представить математическое выражение в виде целочисленной матрицы, что дает больше гибкости при программировании данного метода. В данной работе не производится никаких аналитических изменений в исходной постановке задачи, однако в данном случае выбраны неопределенные параметры объекта, значения которых задаются множеством точек в заданном интервале. Данное преобразование придает свойство робастности системе управления. При этом функционалы вычисляются для каждой точки из множества неопределенных параметров, а результирующим считается сумма функционалов по

всем точкам каждого параметра. В данной работе применяется параллельная реализация метода сетевого оператора. Такая реализация позволяет рассмотреть большее количество возможных решений без существенного увеличения временных затрат, а соответственно, шанс найти необходимое решение возрастает. Также в данной реализации предоставляется возможность снизить время работы всего алгоритма за счет распараллеливания процесса вычисления функционалов.

4. Принципы распараллеливания

Поисковая часть метода сетевого оператора основана на генетическом алгоритме. Существует несколько моделей распараллеливания генетических алгоритмов [8 - 10], из них три основных, остальные являются их расширениями и комбинациями:

1. Master-slave (ведущий-ведомый) модель. Модель, при которой ведущий поток (master) обрабатывает единственную популяцию. Таким образом, весь генетический алгоритм производится master-потоком. Ведомые потоки (slave) лишь получают от ведущего потока задания на расчет значений критериев (функционалов). В силу существования ведущего и ведомых потоков такая модель часто носит название master-slave модели. Данная модель особенно эффективна, если критерии качества описываются сложными математическими выражениями, то есть затрачивается большое количество времени на их вычисление;

2. Клеточная модель. В данной модели популяция разбивается на мелкие части (клетки). Такая модель часто именуется выражением «мелкозернистый параллелизм» (fine grained parallelism). Основной идеей данной модели является разделение популяции на мелкие группы хромосом (обычно по одной или две). Таким образом, хромосомы могут взаимодействовать только внутри своей подгруппы. Данная модель изначально разрабатывалась для многопроцессорных систем (как правило, с очень большим количеством процессоров), например, современных видеокарт и суперкомпьютеров;

3. Распределенная модель. В данной модели популяция также разбивается на части, но не настолько мелкие, как в клеточной модели. Поэтому данная модель также носит название «крупнозернистый параллелизм» (coarse grained parallelism). Популяции могут быть разделены как случайным образом, так и по какому-либо признаку. Субпопуляции, полученные после разбивки основной популяции, могут обмениваться своими хромосомами (процесс миграции). На процесс миграции могут быть наложены ограничения, при которых хромосома может мигрировать только в определенные субпопуляции. Субпопуляции часто называют островами, из-за чего сама модель часто именуется также и островным [5] подходом или островной моделью параллелизма.

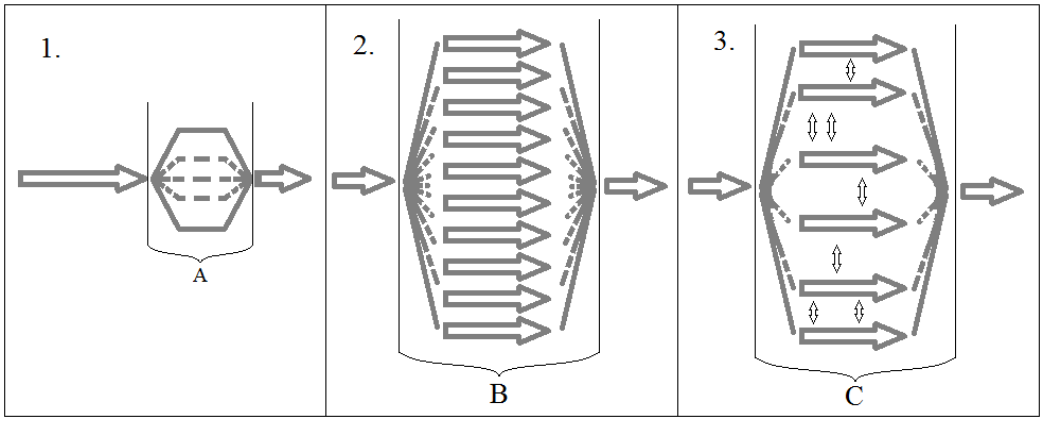

Рис. 1. Три подхода распараллеливания. **A** – часть алгоритма, вычисляющая значения функционалов; **B** – часть алгоритма, в которой выполняются генетические алгоритмы в каждой субпопуляции в отдельной потоке; **C** - часть алгоритма, в которой выполняются генетические алгоритмы в каждой субпопуляции в отдельном потоке, при этом они могут обмениваться хромосомами.

Существуют гибридные (смешанные) модели распараллеливания генетических алгоритмов [8], которые основаны на одной из трех вышеописанных моделей, однако вбирают в себя

некоторые принципы остальных моделей. Например, модель ведущий-ведомый разделяют на синхронную и асинхронную. Разница заключается в наличии возможности продолжения вычислений ведущим потоком без ожидания ведомых. Распределенную модель с миграциями иногда разделяют на островную модель и модель ступенек. Разница состоит в том, что при островной модели миграции хромосом могут происходить между любыми субпопуляциями (мигранты, путешествующие по островам), а при модели ступенек миграция происходит только между соседними субпопуляциями.

Иногда процедуру миграции полностью исключают из алгоритма, но вместо этого включают понятие пересекающихся областей субпопуляций. Тогда в процессе исполнения алгоритма в операции скрещивания внутри одной субпопуляции могут попасть хромосомы из области общего доступа. Так решается проблема обмена между субпопуляциями.

Существует так называемые грязные генетические алгоритмы, для которых также существуют параллельные реализации. Чаще всего грязный генетический алгоритм имеет два цикла: внешний и внутренний. Основная параллельная область исполняется на стадии внутреннего цикла, но внешний цикл также поддается распараллеливанию, однако это зависит от реализации алгоритма.

Самым простым подходом распараллеливания генетических алгоритмов считается подход изолированного исполнения. Алгоритм реализуется так, что на каждом отдельном процессоре или ядре полностью исполняется последовательный алгоритм. Таким образом, на многопроцессорной системе запускается сразу несколько последовательных алгоритмов, которые никак не связаны друг с другом. Соответственно, после того, как каждый процессор завершит вычисления, будет получено большее множество возможных решений, чем при работе лишь одной копии алгоритма.

5. Синтез системы управления посадкой КА на поверхность Луны

Эксперимент проводился на модели КА, движение которого задано системой из пяти дифференциальных уравнений (1-4):

$$\begin{aligned}
\dot{x}_0 &= \frac{g_E \cdot (P_{0c} + u_1) \cdot \cos(u_0 - x_1)}{x_4} - g_{Mh} \cdot \cos x_1 \\
\dot{x}_1 &= \frac{g_E \cdot (P_{0c} + u_1) \cdot \sin(u_0 - x_1)}{x_4} - g_{Mh} \cdot \sin x_1 \\
\dot{x}_2 &= \frac{x_0 \cdot \cos x_1}{1000} \\
\dot{x}_3 &= \frac{x_0 \cdot \sin x_1}{(R_M + x_2) \cdot 1000} \\
\dot{x}_4 &= -\frac{(P_{0c} + u_1)}{P_{udc}}
\end{aligned}, \qquad (11)$$

$$g_{Mh} = g_M \cdot \left(\frac{R_M}{R_M + x_2} \right)^2, \qquad (12)$$

где x_0 – скорость движения КА в текущий момент времени, x_1 – угол наклона траектории КА в текущий момент времени, x_2 – высота полета КА относительно поверхности Луны в текущий момент времени, x_3 – дальность полета КА на текущий момент времени, x_4 – масса КА с учетом топлива, g_E – значение ускорения свободного падения на Земле, g_M – значение ускорения свободного падения на поверхности Луны, g_{Mh} – значение ускорения свободного падения на некоторой высоте над поверхностью Луны, P_{0c} – номинальная тяга двигателя КА (кг), P_{udc} – удельная тяга двигателя КА (с), u_0 – управление углом наклона траектории, $u_0(x_1, x_2) \in [0; \pi]$, u_1 – управление тягой двигателя КА, $u_1(x_1, x_2) \in [-80; 80]$.

Терминальные условия заданы следующим образом:

- Конечная скорость: 0 м/с;

- Конечная высота: 1.5 км.

Начальные условия заданы следующим образом:

Таблица 1. Начальные условия

Угол наклона траектории	Начальная масса с учетом топлива	Начальная скорость	Начальная высота	Начальное отклонение угловой дальности
$\theta^-(0) = -1.65$ рад $\theta^+(0) = -1.55$ рад	$m(0) = 940$ кг	$V(0) = 1689$ м/с	$h^-(0) = 17$ км $h^+(0) = 20$ км	$\varphi(0) = 0$ рад

Функционал качества и функционал цели управления для данной системы заданы в следующем виде:

$$J_1 = \sum_{i=1}^{P} \left(\left| h_f - h(t_f) \right| + \nu_1 \cdot \left| \varphi - \varphi(t_f) \right| \right) \to 0, \tag{13}$$

$$J_2 = \sum_{1}^{k} \left(\left| V_f - V(t_f) \right| + \nu_2 \cdot \left| \varphi_f - \varphi(t_f) \right| \right) \to 0, \tag{14}$$

где ν_1, ν_2 - весовые коэффициенты. В отличие от [2] в данном эксперименте использовался функционал с учетом дальности отклонения от планируемой точки посадки.

Для данного эксперимента в качестве неопределенных параметров выбраны начальная высота, с которой начинается процесс посадки и начальный угол наклона КА. Начальная высота рассчитывается как точка перицентра Луны (периселений), но для обеспечения робастности системы рассматривается конечное множество точек, в котором может находиться КА. Начальный угол наклона КА может также отличаться от заданного и, соответственно, задается конечным множеством точек:

$$y_1 = \left\{ y_1^1(t_0), ..., y_1^l(t_0) \right\} \Leftrightarrow \left\{ h^1(t_0), ..., h^l(t_0) \right\}, \tag{15}$$

$$y_2 = \left\{ y_2^1(t_0), ..., y_2^l(t_0) \right\} \Leftrightarrow \left\{ \theta^1(t_0), ..., \theta^l(t_0) \right\}, \tag{16}$$

где y_1 – отклонение по начальной высоте, y_2 – отклонение по начальному углу. Величина l – выбирается в зависимости от величины интервала $\left| h^1(t_0) - h^l(t_0) \right|$ или $\left| \theta^1(t_0), ..., \theta^l(t_0) \right|$ и от необходимой степени робастности по определенному параметру. Однако необходимо учесть, что высокая дискретность данных интервалов влечет за собой увеличение времени расчетов, так как расчеты функционалов производятся для каждой комбинации неопределенных параметров. Таким образом, для случая, когда $P = 3$ и $l = 3$, функционалы будут рассчитаны $P \times l = 9$ раза.

В данном эксперименте применяется гибридная модель распараллеливания - комбинация модели ведущий-ведомый и островной модели распараллеливания. Выбор модели ведущий-ведомый обусловливается необходимостью вычислять функционалы для нескольких точек из области неопределенных параметров. Выбор островной модели обусловливается содержанием большого количества хромосом в каждой популяции. Для достижения большей эффективности распараллеливания применяется комбинация данных моделей. Перед началом работы алгоритма исследователю предлагается выбрать количество процессоров, используемых на распределение по островам. Учитывая данное значение, программа автоматически вычисляет количество оставшихся процессоров, которое используется для вычисления функционалов. Также перед началом работы исследователь задает количество поколений, пар в поколении, эпох (определяет размер цикла поколений, после которого происходит отбор лучшего решения от каждого острова). По прохождении эпохи каждый остров принимает отобранную лучшую хромосому в качестве базисной, и дальнейшая работа происходит именно с ней (миграция хромосом). Каждый остров имеет в своем распоряжении определенное количество процессоров, которые используются для расчета функционалов. Параллельность алгоритма расчета функционалов заключается в независимом расчете значения функционала для каждой точки из множества точек каждого неопределенного параметра. Работа организована по

алгоритму, при котором поток, освободившись от вычисления функционала при определенной точке неопределенных параметров, тут же переходит на следующую точку. Соответственно, в таком режиме запускаются сразу несколько потоков. Каждое полученное значение функционала автоматически суммируется к значению функционала для каждой хромосомы. Таким образом, такой процесс заметно ускоряет процедуру расчета функционалов. Параметры генетического алгоритма выглядят следующим образом:

- размер эпохи (количество поколений, после которых происходит обмен): 25;
- количество пар в поколении (количество сгенерированных пар за поколения): 128;
- общее количество поколений: 500;
- размерность каждой популяции (количество хромосом в каждой популяции):1000;

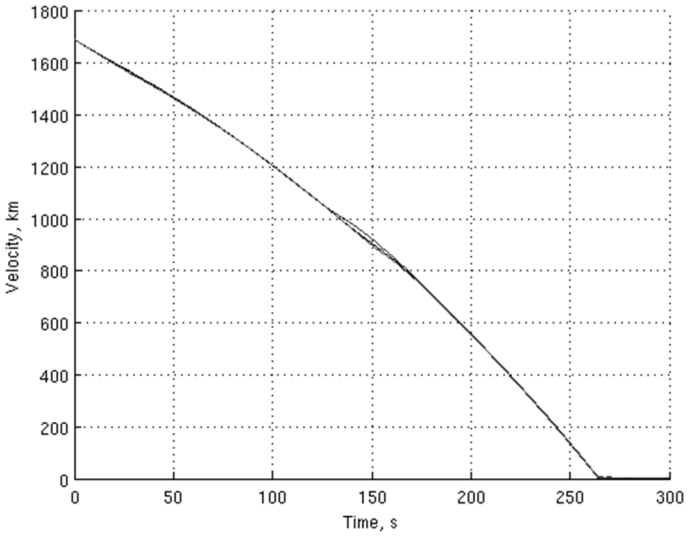

Рис. 2. Изменение скорости

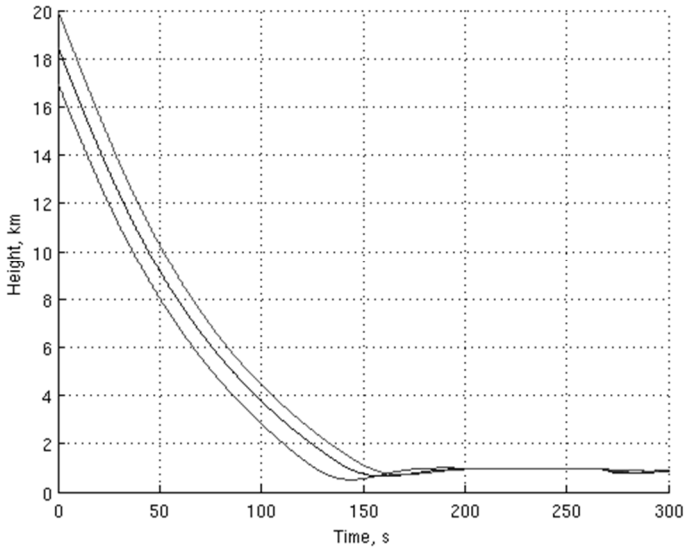

Рис. 3. Изменение высоты при разных НУ

На рис. 2 и рис. 3 показаны графики изменения скорости и высоты при разных начальных условиях: $h^1(0) = 17$ км, $h^2(0) = 18.5$ км, $h^3(0) = 20$ км. Таким образом, КА выходит на необходимую высоту с разных начальных условий и достигает на ней нулевую скорость.

При тех же параметрах алгоритма был проведен сравнительный анализ времени вычислений, результаты которых представлены ниже.

Рис. 4. Зависимость времени от количества ведомых ядер

На рис. 4 показан график зависимости времени выполнения алгоритма от количества ведомых ядер (выделенных на вычисление функционалов) при одних начальных параметрах. По графику видно, что при увеличении количества ведомых ядер время расчета уменьшается, но лишь до определенного момента. В частности, при выделении 16 ядер на вычисление функционалов, время начинает расти, из-за выделения большего времени на создание и уничтожение потоков. Это объясняется тем, что процедура расчета функционалов вызывается большое количество раз в процессе работы всего алгоритма. Таким образом, общее служебное время (время на создание, уничтожение и коммуникацию между потоками), начинает перекрывать время, выигранное при параллельном вычислении. Однако, при увеличении размерностей параметров генетического алгоритма, выигрыш от распараллеливания перекроет служебное время.

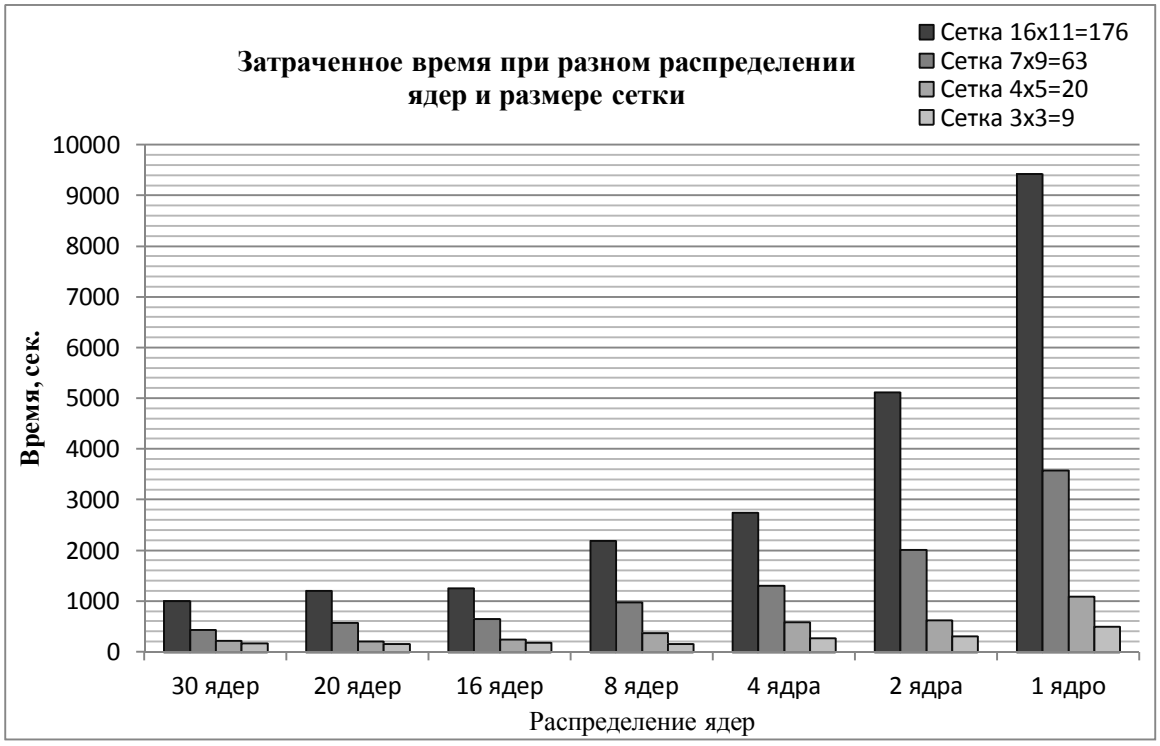

Рис. 5. Зависимость времени от размерности сетки неопределенных параметров и количества ведомых ядер

На рис. 5 показан график зависимости времени расчета от размерности сетки неопределенных параметров и количества ведомых ядер (выделенных на вычисление функционалов). По графику видно, что разница между временем однопоточного и многопоточного вычислений растет с увеличением размерности сетки параметров. Для проведения сравнительного анализа времени использовались следующие параметры генетического алгоритма и распределения ядер:

- размер эпохи (количество поколений, после которых происходит обмен): 10;
- количество пар в поколении (количество сгенерированных пар за поколения): 32;
- общее количество поколений: 16;
- размерность каждой популяции (количество хромосом в каждой популяции): 32;
-

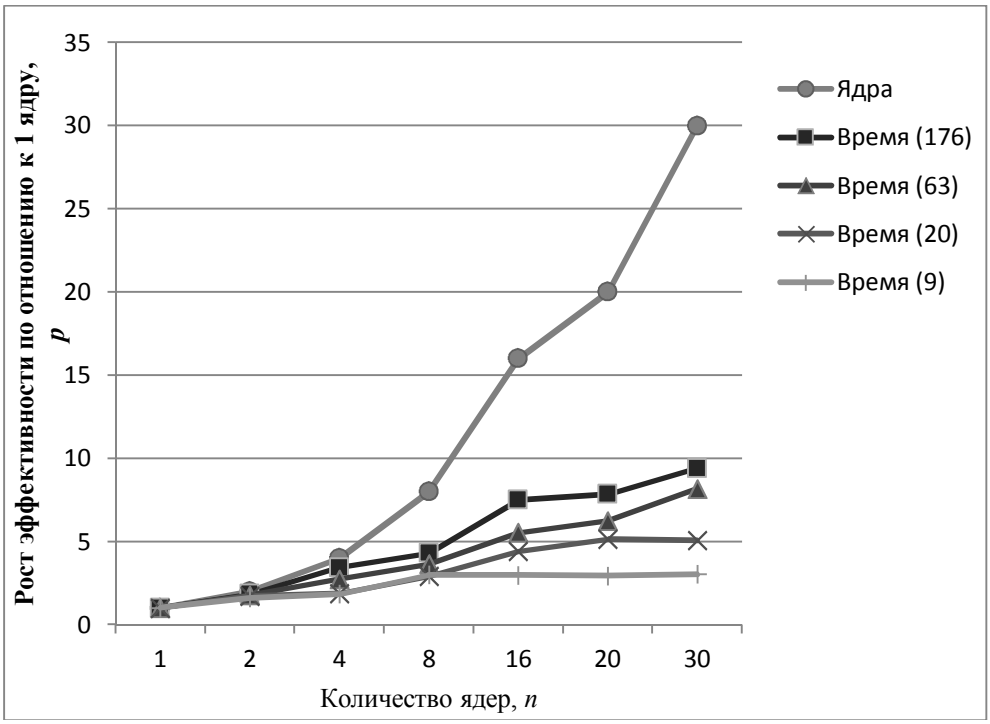

Рис. 6. Рост эффективности по отношению к 1 ядру

На рис. 6 показан график, отображающий рост эффективности вычисления по времени и количеству ядер по отношению к показателям при вычислении на одном ядре:

$$p_i = \frac{t_i}{t_1}, i = \overline{1,n}, \tag{17}$$

где p – ускорение для разных размерностей сетки неопределенных параметров, t_i – время вычисления на i ядрах. Значения вычисляются отношением показателей при n ядрах (n = 2, 4, 8, 16, 20, 30) к показателям при вычислении на одном ядре. По данному графику можно сделать вывод, что эффективность распараллеливания повышается при увеличении размерности сетки, однако при малом количестве ядер (n = 2, 4) скорость роста эффективности по времени сопоставима со скоростью роста производительности машины по ядрам.

На рис. 7 показан график снижения эффективности по времени по отношению к количеству ядер. За начало взята точка 1 (1 ядро). Таким образом, при увеличении количества ядер эффективность по времени снижается (что также видно по рис. 6). Скорость снижения эффективности увеличивается с уменьшением размерности сетки неопределенных параметров.

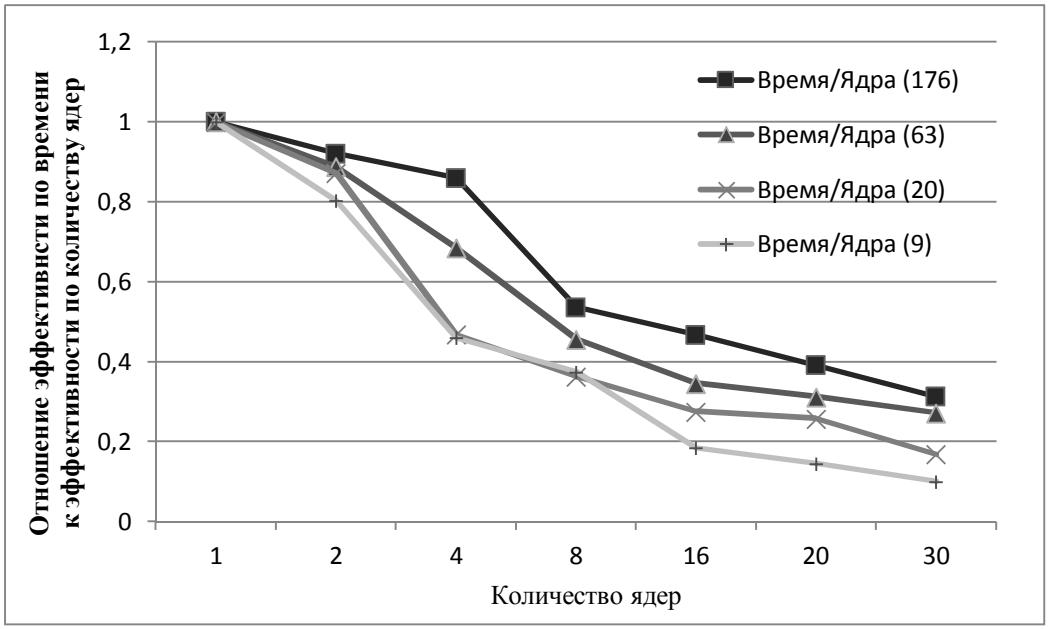

Рис. 7. Отношение эффективности по времени к эффективности по количеству ядер

Для повышения эффективности алгоритма решения задачи численного синтеза системы управления посадкой космического аппарата был разработан параллельный метод сетевого оператора, позволяющий рассмотреть большее количество возможных решений, чем при вычислении с использованием одного ядра. Соответственно, это увеличивает шансы нахождения подходящего решения. Разработанный метод также позволяет ускорить время работы алгоритма путем распараллеливания области вычисления функционалов. Разработанный алгоритм позволяет балансировать нагрузку на ядра процессоров, распределяя ее между независимыми базисными потоками и вычислением функционалов для каждой точки из множества неопределенных параметров. По результатам проведенных вычислений представлены графики, характеризующие эффективность разработанного метода по временным параметрам. Из полученных результатов можно сделать вывод, что эффективность вычислений выше при большей размерности сетки неопределенных параметров. Рост эффективности замедляется с ростом количества используемых ядер в параллельной области, вычисляющей функционалы. Это связано с б**о**льшими временными затратами на создание и уничтожение параллельных потоков, так как процедура вызова функционалов вызывается большое количество раз.

Литература

1. Дивеев А.И., Метод сетевого оператора, Москва, ВЦ РАН, 2010, 178 с.
2. Дивеев А.И., Численный метод сетевого оператора для синтеза системы управления с неопределенными начальными значениями, Известия РАН, Теория и системы управления, № 2, 2012, с. 63-78.
3. Дивеев А.И., Уваров А.С., Хамадияров Д.Б., Параллельный алгоритм численного синтеза робастной системы управления на основе метода сетевого оператора, труды XI международного симпозиума «Интеллектуальные системы INTELS’ 2014», Москва, РУДН 30.06 – 04.07 2014 г., с. 88-91.
4. Карпенко А.П., Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов, Информационные технологии, Новые технологии, №7, 2012, с.2-32.
5. Курейчик В.М., Кныш Д.С.: Параллельный генетический алгоритм. Модели и проблемы построения, Технологический Институт ЮФУ Таганрог, 2010, с. 1-3.

6. Diveev A., Khamadiyarov D., Shmalko E., Sofronova E.: Intellectual Evolution Method for Synthesis of Mobile Robot Control System, Proceeding of IEEE Congress on Evolutionary Computation (CEC), Cancún, México, 2013, pp. 24-31.
7. Koza J.R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, Massachusetts, London, MA: MIT Press, 1992, 819 p.
8. Luque G., Alba E., Parallel Genetic Algorithms, 2011, 367 p.
9. Nowostawski M., Riccardo P., Parallel Genetic Algorithm Taxonomy, Proceedings of Knowledge-Based Intelligent Information Engineering Systems (KES), 13.05.1999.
10. Stender J., Parallel Genetic Algorithm: Theory & Applications, 1994, 223 p.