РОССИЙСКАЯ АКАДЕМИЯ НАУК
УРАЛЬСКОЕ ОТДЕЛЕНИЕ
ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ

АЛГОРИТМЫ И ПРОГРАММНЫЕ СРЕДСТВА ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Сборник научных трудов

Выпуск 3

ЕКАТЕРИНБУРГ
1999

УДК 519.688+517.9

Алгоритмы и программные средства параллельных вычислений: [Сб.науч.тр.]. Екатеринбург: УрО РАН, 1999. Вып. 3. ISBN 5–7691–0985–8.

Сборник посвящен вопросам организации параллельных вычислений и параллельным алгоритмам, рассмотренным на семинаре ИММ УрО РАН по параллельным вычислениям.

Материалы могут быть рекомендованы специалистам, разрабатывающим параллельные алгоритмы, а также разработчикам программного обеспечения для параллельных машин.

Ответственный редактор к.ф.-м.н. О.Н.Ульянов

Издание подготовлено в пакете CyrTUG-emTex с использованием кириллических шрифтов семейства LH

ISBN 5–7691–0985–8

$\text{А}\frac{\text{ПРП} - 1999 - 123(99)}{8\text{П6(03)}1998}\text{ПВ} - 1999$

Уральское отделение Российской академии наук
Институт математики и механики

Алгоритмы и программные средства параллельных вычислений — 1999. Вып. 3

О РАЗРАБОТКЕ ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ РЕШЕНИЯ ЛИНЕЙНЫХ ДИФФЕРЕНЦИАЛЬНЫХ ИГР *

С. С. Кумков

Описываются алгоритмы, входящие в состав программы решения линейных дифференциальных игр: для построения множеств уровня функции цены, для конструирования оптимальных стратегий игроков, для обнаружения и классификации сингулярных поверхностей. Приводится краткое описание алгоритма параллельной программы. Раскрываются проблемы, возникшие при разработке параллельной программы. Рассматриваются библиотеки процедур, написанные для решения этих проблем.

1. Основные алгоритмы

Рассматривается линейная антагонистическая дифференциальная игра [1]

$$\dot{x} = A(t)x + B(t)u + C(t)v, \qquad (1)$$
$$x \in R^n, \quad u \in P, \quad v \in Q, \quad \varphi(x_i(T), x_j(T))$$

с фиксированным моментом окончания T и выпуклой функцией платы φ, зависящей от двух компонент x_i, x_j фазового вектора.

*Работа выполнена при поддержке Российского фонда фундаментальных исследований, гранты № 97–01–00672 и № 99–07–90441.

Первый (второй) игрок распоряжается управлением u (v), выбирая его из выпуклого компакта P (Q) так, чтобы минимизировать (максимизировать) значение функции φ в момент T.

Пусть $X_{i,j}(T,t)$ — матрица, составленная из двух строчек фундаментальной матрицы Коши, соответствующих координатам x_i, x_j. Тогда замена переменных $y(t) = X_{i,j}(T,t)x(t)$ обеспечивает [1] переход к эквивалентной дифференциальной игре второго порядка по фазовой переменной.

Для линейной дифференциальной игры (1) в начале 80-х годов в Институте математики и механики Уральского отделения Академии наук были разработаны попятные алгоритмы построения множеств уровня функции цены [2, 3].

1.1. Построение множеств уровня функции цены

Ниже приводится алгоритм построения множеств уровня функции цены из статьи [4].

Попятная процедура. Положим, что от игры (1) с функцией платы φ, зависящей от двух компонент фазового вектора, уже сделан переход к эквивалентной игре

$$\begin{gathered} \dot{y} = D(t)u + E(t)v, \\ y \in R^2, \quad u \in P, \quad v \in Q, \quad \varphi(y(T)) \\ D(t) = X_{i,j}(T,t)B(t), \quad E(t) = X_{i,j}(T,t)C(t). \end{gathered} \tag{2}$$

Пусть на промежутке $[0,T]$ задана последовательность моментов $t_i : \ t_N = T, \ldots, t_i = t_{i+1} - \Delta, \ldots, t_0 = 0$, разбивающая его с шагом Δ. Требуется найти сечение множества уровня $W_c(t_i) = \{y \in R^2 : V(t_i, y) \le c\}$ функции цены V для заданного значения параметра c.

Заменим динамику (2) кусочно-постоянной динамикой

$$\begin{gathered} \dot{y} = \mathbf{D}(t)u + \mathbf{E}(t)v, \\ \mathbf{D}(t) = D(t_i), \quad \mathbf{E}(t) = E(t_i), \quad t \in [t_i, t_{i+1}). \end{gathered} \tag{3}$$

Вместо множеств P и Q будем рассматривать их выпуклые многогранные аппроксимации $\mathbf{P}$, $\mathbf{Q}$. Также пусть $\hat{\varphi}$ — аппроксимирующая функция платы, такая что для любого значения c ее

множество уровня $\mathbf{M}_c = \{y : \hat{\varphi}(y) \le c\}$, соответствующее этому значению, является выпуклым многоугольником.

Аппроксимирующая игра (3) выбирается так, что на каждом шаге $[t_i, t_{i+1}]$ попятной процедуры мы имеем дело с игрой с простыми движениями, выпуклыми многогранными ограничениями на управления игроков и выпуклым многоугольным целевым множеством. Положим $\mathbf{W}_c(t_N) = \mathbf{M}_c$. Далее найдем множество разрешимости $\mathbf{W}_c(t_{N-1})$ игры с простыми движениями, затем $\mathbf{W}_c(t_{N-2})$ и т.д. В качестве результата мы получим набор выпуклых множеств, которые приближают в метрике Хаусдорфа множество уровня $\mathbf{W}_c$ функции цены игры (2).

Обозначим $\mathcal{P}(t_i) = -D(t_i)\mathbf{P}$, $\mathcal{Q}(t_i) = E(t_i)\mathbf{Q}$. Опорная функция $l \to \rho(l, \mathbf{W}_c(t_i))$ множества $\mathbf{W}_c(t_i)$ является выпуклой оболочкой [5] функции

$$\gamma(l, t_i) = \rho(l, \mathbf{W}_c(t_{i+1})) + \Delta\rho(l, \mathcal{P}(t_i)) - \Delta\rho(l, \mathcal{Q}(t_i)).$$

Функция $\gamma(\cdot, t_i)$ является положительно-однородной и кусочно-линейной. При этом свойство локальной выпуклости этой функции может нарушаться лишь на границе конусов линейности функции $\rho(\cdot, \mathcal{Q}(t_i))$, т.е. на границе конусов, порождаемых нормалями к соседним ребрам многоугольника $\mathcal{Q}(t_i)$.

Алгоритм построения выпуклой оболочки. В дальнейшем аргумент t_i в записи функции γ будет опускаться для упрощения записи. Вместо множеств $\mathcal{P}(t_i)$, $\mathcal{Q}(t_i)$ будем писать просто $\mathcal{P}$, $\mathcal{Q}$. Также вместо $\mathbf{W}_c(t_i)$ и $\mathbf{W}_c(t_{i+1})$ будем писать $\mathbf{W}_c^i$ и $\mathbf{W}_c^{i+1}$ соответственно.

Конусы линейности γ определяются нормалями к ребрам многоугольников $\mathbf{W}_c^{i+1}$, $\mathcal{P}$, $\mathcal{Q}$. Собрав эти нормали и упорядочив их по часовой стрелке, получим набор векторов L. Набор значений $\gamma(l)$ функции γ на векторах $l \in L$ обозначим Ф. Наборы L, Ф полностью описывают функцию γ.

Множество нормалей к многоугольнику $\mathcal{Q}$, упорядоченное по часовой стрелке, обозначим через S. Набор S будем называть набором "подозрительных" векторов. Это название связано с тем, что функция γ наверняка выпукла на конусах, внутренность которых не содержит нормалей к $\mathcal{Q}$, и нарушение локаль-

ной выпуклости может происходить лишь только на конусах, содержащих внутри себя по крайней мере одну нормаль к многоугольнику $\mathcal{Q}$.

Положим $L^{(1)} = L$, $\Phi^{(1)} = \Phi$, $S^{(1)} = S$. Шаг с номером $(k+1)$ процесса овыпукления заключается в замене наборов $L^{(k)}$, $\Phi^{(k)}$ некоторыми другими наборами $L^{(k+1)} \subset L^{(k)}$, $\Phi^{(k)} \subset \Phi^{(k+1)}$. При этом $S^{(k)}$ также подменяется новым набором $S^{(k+1)}$.

Опишем один шаг процесса овыпукления. Положим, что угол между любыми двумя соседними векторами из набора $L^{(k)}$, отсчитываемый по часовой стрелке, меньше π. Пусть $l \to \gamma^{(k)}(l)$ — кусочно-линейная функция, определяемая наборами $L^{(k)}$, $\Phi^{(k)}$. Поскольку

$$L^{(k)} \subset L^{(k-1)} \subset \cdots \subset L^{(1)} \quad \text{и} \quad \Phi^{(k)} \subset \Phi^{(k-1)} \subset \cdots \subset \Phi^{(1)},$$

то для любого вектора $\bar{l} \in L^{(k)}$ значение $\gamma^{(k)}(\bar{l})$ равно $\gamma(\bar{l})$.

Возьмем вектор $l_* \in S^{(k)}$ и проверим локальную выпуклость функции $\gamma^{(k)}$ на конусе, порожденном вектором l_* и двумя соседними к нему векторами l_- and l_+, взятыми по часовой стрелке и против нее из набора $L^{(k)}$. Другими словами, проверим, является ли неравенство $l_*' y \le \gamma(l_*)$ существенным в тройке неравенств $l_-' y \le \gamma(l_-)$, $\quad l_*' y \le \gamma(l_*)$, $\quad l_+' y \le \gamma(l_+)$. Если эта система неравенств совместна, то (в силу упорядоченности векторов $l_-,\ l_*,\ l_+$) лишь средний из них может быть несущественным.

Алгоритм проверки существенности: найдем точку y_*, в которой пересекаются прямые $l_-' y = \gamma(l_-)$ и $l_*' y = \gamma(l_*)$. Затем проверим неравенство $l_+' y_* < \gamma(l_+)$. Если оно выполняется, то локальная выпуклость имеет место (среднее неравенство существенно). В противном случае локальная выпуклость отсутствует (среднее неравенство не является существенным).

В первом случае вектор l_* исключается из набора $S^{(k)}$. Полученное множество обозначается $S^{(k+1)}$. При этом $L^{(k+1)} = L^{(k)}$, $\Phi^{(k+1)} = \Phi^{(k)}$.

Во втором случае возможны две ситуации. Обозначим через α угол между векторами l_- to l_+, отсчитанный по часовой стрелке:

• $\alpha < \pi$. Вектор l_* извлекается из набора $S^{(k)}$, и, одновременно, векторы l_- and l_+ включаются туда. (При этом один из них или оба уже могут присутствовать там.) Полученный таким образом новый набор "подозрительных" векторов обозначается $S^{(k+1)}$. Набор $L^{(k+1)}$ получается из $L^{(k)}$ исключением вектора l_*. Аналогично, при переходе от $\Phi^{(k)}$ к $\Phi^{(k+1)}$ оттуда выкидывается значение $\gamma^{(k)}(l_*) = \gamma(l_*)$;

• $\alpha \geq \pi$. Данное неравенство означает, что рассматриваемая тройка неравенств несовместна. И, стало быть, выпуклая оболочка функции γ не существует, т.е. $\mathbf{W}_c^i = \emptyset$. Дальнейшие построения прекращаются. (Если $\alpha = \pi$, то возможно, что $\mathbf{W}_c(t_i)$ является вырожденным многоугольником, т.е. $\mathbf{W}_c(t_i)$ — точка или отрезок. В этом случае дальнейшие построения также прекращаются.)

Тем самым завершено описание одного шага овыпукления. Процесс завершается на шаге с номером j, когда впервые $S^{(j)} = \emptyset$, т.е. когда в первый раз множество "подозрительных" нормалей становится пустым. Это означает, что функция $\gamma^{(j)}$, определяемая наборами $L^{(j)}$ и $\Phi^{(j)}$, является локально выпуклой всюду. Таким образом, она и является выпуклой оболочкой функции γ. Другой вариант завершения: на каком-то шаге угол α между векторами l_- и l_+ становится больше или равным π после исключения проверяемого вектора l_* из набора "подозрительных" векторов. Это означает, что $\mathbf{W}_c^i = \emptyset$.

1.2. Конструирование оптимальных стратегий

Опишем алгоритм построения оптимальной стратегии первого игрока в случае, когда его управляющий параметр — скаляр (P — отрезок $[-\mu, \mu]$). В этих условиях существует [6] универсальная оптимальная стратегия первого игрока, т.е. стратегия, оптимальная для всех начальных позиций (t_0, x_0). Стратегия является устойчивой по отношению к погрешностям аппроксимации (3). Реализовать стратегию можно при помощи некоторой специальной поверхности $S^{(1)}$, разделяющей трехмерное пространство (t, y_1, y_2) эквивалентной игры (2) на две части $S_+^{(1)}$ и

$S_-^{(1)}$. Если в момент t состояние системы (1) есть $x(t)$ и соответствующая ему позиция $(t, y(t))$ эквивалентной игры принадлежит $S_+^{(1)}$, то следует взять $U^0(t, x(t)) = \mu$. Если $(t, y(t))$ принадлежит $S_-^{(1)}$, то $U^0(t, x(t)) = -\mu$. В случае $(t, y(t)) \in S^{(1)}$ значение $U^0(t, x(t))$ можно брать произвольным из $[-\mu, \mu]$. Таким образом, поверхность $S^{(1)}$ является поверхностью переключения управления, а ее сечение при фиксированном t — линией переключения.

Поскольку реализация во времени стратегии первого игрока происходит в дискретной схеме управления, то сечения поверхности переключения достаточно построить лишь для моментов выбранного в дискретной схеме разбиения промежутка времени игры. Условимся, что это разбиение принадлежит разбиению, используемому в попятной процедуре построения сечений моста $\mathbf{W}_c$ в аппроксимирующей игре (3) с целевым множеством M_c.

Пусть t — произвольный момент, для которого нужно построить линию переключения. Выбрав достаточно большое число $\overline{\overline{c}}$, зададим разбиение $\Omega = \{0 < c_1 < c_2 < \cdots < c_r = \overline{\overline{c}}\}$ отрезка $[0, \overline{\overline{c}}]$. Для каждого c_k строим сечение $\mathbf{W}_{c_k}(t)$ моста $\mathbf{W}_{c_k}$ при помощи попятной процедуры. Пусть $\overline{c}(t) = c_p \in \Omega$ — наименьшее c_k, для которого $\mathbf{W}_{c_k}(t) \neq \emptyset$. Рассматриваем сечения $\mathbf{W}_{c_k}(t)$ для $c_k \geq \overline{c}(t)$. Каждое из множеств $\mathbf{W}_{c_k}(t), k \in \overline{p, r}$, замкнуто, выпукло, ограничено. Обходя его границу, найдем точку β_k (α_k), где скалярное произведение вектора $D(t) = X_{i,j}(T, t)B$ на вектор внешней нормали к границе $\mathbf{W}_{c_k}(t)$ меняет знак с плюса на минус (с минуса на плюс). Перебирая $k \in \overline{p, r}$, получим наборы $\beta_p, \beta_{p+1}, \ldots, \beta_r$ и $\alpha_p, \alpha_{p+1}, \ldots, \alpha_r$. Соединив последовательно точки этих наборов, а также β_p с α_p, построим ломаную. Ее и следует взять в качестве линии переключения $S^{(1)}(t)$. Степень приближения к "идеальной" линии переключения тем выше, чем меньше диаметр выбранного разбиения отрезка $[0, \overline{\overline{c}}]$. Построение линии переключения качественно поясняется на рис. 1, где направление вектора $D(t)$ показано стрелкой.

Вообще говоря, $S^{(1)}(t)$ не разбивает все пространство на две части. Кривая $S^{(1)}(t)$ однозначно разделяет на две части лишь полосу $\Lambda(t)$ на плоскости (y_1, y_2), составленную из всех прямых,

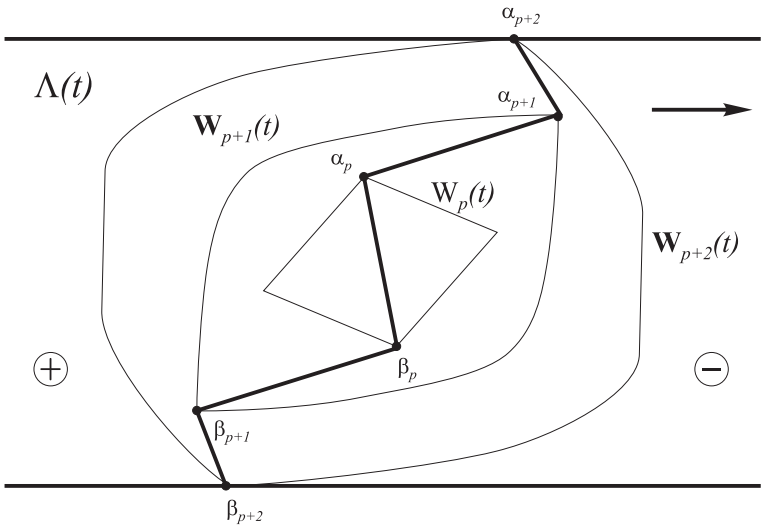

Рис. 1. Построение линии переключения

проходящих через $\mathbf{W}_{\bar{\bar{c}}}(t)$ параллельно вектору $D(t)$. Однако выбором $\bar{\bar{c}}$ можно сделать полосу $\Lambda(t)$ достаточно большой, чтобы любое разумное моделируемое движение $y(t)$ системы (2) для любого момента t находилось в полосе $\Lambda(t)$. Обозначим ту половину $\Lambda(t)$, куда направлен вектор $D(t)$, через $S_-^{(1)}(t)$, противоположную — $S_+^{(1)}(t)$ (на рис. 1 эти части обозначены соответственно знаками минус и плюс). Содержательно стратегию U^0 первого игрока можно пояснить следующим образом: выбирается такое управление u, которое максимально "подтягивает" положение системы к линии $S^{(1)}(t)$ — в сторону уменьшения c.

Универсальная оптимальная стратегия V^0 второго игрока строится аналогичным образом с заменой вектора $D(t)$ на $E(t) = X_{i,j}(T,t)C$. Знаки областей определяются наоборот: та, куда направлен вектор $E(t)$, назначается областью положительного управления, противоположная — отрицательного. Содержательно это значит, что второй игрок выбирает управление максимально "оттягивающее" положение системы от линии $S^{(2)}(t)$. На самой линии переключения $S^{(2)}(t)$ второго игрока

управление берется либо ν, либо $-\nu$. Запрет на выбор промежуточных значений управляющего воздействия второго игрока приводит к тому, что описанная стратегия не является устойчивой к погрешностям численных построений. Однако эффект от потери устойчивости может проявиться лишь при весьма специальном управлении первого игрока, когда движение попадает на поверхность переключения второго игрока и в дальнейшем скользит вдоль нее. Пренебрегая возможностью специального изощренного поведения первого игрока, считаем численно построенную стратегию второго игрока практически оптимальной.

В случае, когда управление первого или второго игрока векторное, но его компоненты стеснены независимыми ограничениями (т.е. P или Q являются многомерными параллелепипедами), поверхности переключения строятся для каждой компоненты управляющего воздействия независимо. При определенных допущениях стратегия, определенная таким набором поверхностей переключения, является оптимальной.

1.3. Построение сингулярных поверхностей

Под сингулярными поверхностями в теории дифференциальных игр понимаются такие множества в игровом пространстве, на которых оптимальные движения имеют особенности (рассеивание, излом, слияние и т.д.). Классификация сингулярных поверхностей была предложена американским математиком Р.Айзексом в его книге [7]. Необходимые условия, характеризующие тот или иной тип сингулярных поверхностей, изучались А.А.Меликяном [8] и П.Бернаром [9]. Во многих статьях (например в [10, 11]) проводится анализ сингулярных поверхностей, появляющихся в конкретных задачах.

В этом разделе будем считать, что управления обоих игроков являются скалярными, т.е. множества P и Q в системе (1) являются отрезками. Тогда и множества $\mathcal{P}(t_i)$ и $\mathcal{Q}(t_i)$ являются отрезками в пространстве y_1, y_2 в любой момент t_i.

В разработанном методе обнаружения и классификации сингулярных поверхностей базовыми объектами являются сингулярные точки, возникающие на сечениях моста. *Регулярными*

точками являются такие, через которые проходит одно оптимальное движение под воздействием крайних управлений обоих игроков (т.е. управлений, взятых с концов отрезков $\mathcal{P}$ и $\mathcal{Q}$). *Сингулярными* точками называются точки, не являющиеся регулярными.

Сингулярные точки появляются на поверхности сечения в тех местах, где происходит смена управлений игроков. Таким образом, точки переключения, описывавшиеся в предыдущем пункте являются сингулярными. Однако предложенный выше метод не позволяет классифицировать эти точки, т.е. определять характер возникающих в них особенностей оптимальных движений.

Для обнаружения сингулярных точек на очередном сечении моста и их классификации используется информация, накапливаемая в процессе построения выпуклой оболочки функции γ. А именно, отслеживаются нормали, взятые с многоугольника $\mathcal{P}(t_i)$ управления первого игрока ($\mathcal{P}$-нормалей) и границы зон, в которых производилось "подправление" нарушений выпуклости функции γ. (Нарушение выпуклости функции γ может происходить около направлений, взятых с многоугольника $\mathcal{Q}(t_i)$ управления второго игрока — $\mathcal{Q}$-нормалей.)

Возможны три ситуации:

• зона "подправления" не содержит $\mathcal{P}$-нормалей. Тогда возникает особенность рассеивания — точка, которая не принимает оптимальных движений, но порождает два оптимальных движения при различных крайних управлениях второго игрока;

• $\mathcal{P}$-нормаль лежит "вдали" от зоны "подправления". Тогда возникает ребро "переключения" первого игрока, разделяющее зоны с разными крайними управлениями первого игрока. На самом ребре оптимальное управление первого игрока может быть некрайним, т.е. выбираться из внутренности отрезка $\mathcal{P}$;

• $\mathcal{P}$-нормаль является граничной нормалью зоны "подправления". В этом случае возникает ситуация экивокальности; т.е. появляется точка, принимающая оптимальные движения и порождающая два оптимальных движения. Причем одно из них происходит при крайних управлениях обоих игроков, а другое — при крайнем управлении второго игрока, но некрайнем первого.

После построения очередного сечения (после завершения процесса построения выпуклой оболочки функции γ) на нем находятся и классифицируются сингулярные точки. По завершении счета моста информация о сингулярных точках на его сечениях записывается на диск и может быть использована в дальнейшем. Таким образом, к моменту окончания работы этой программы никаких сингулярных поверхностей и линий еще не сформировано. Линии и поверхности строятся позже в программе визуализации. Вообще задача восстановления сингулярных поверхностей по отдельным насчитанным точкам и обоснование корректности такого восстановления сложны сами по себе. В данный момент принята следующая методика визуального представления полученных результатов.

Вначале из отдельных точек, снятых с одного моста, формируются сингулярные линии. При этом на каждом из просчитанных мостов точки соответствующих типов с соседних сечений соединяются между собой. Однако возникает проблема соединения точек разного типа (например там, где линия переключения за первого игрока, сливаясь с линией рассеивания за второго игрока, переходит в экивокальную).

Затем производится соединение сингулярных линий, снятых с разных мостов. Здесь используется та же идеология, что и при формировании линий: соединяются точки, имеющие одинаковый тип сингулярности. Возникает та же проблема соединения разнотипных точек. Кроме того, поскольку разные мосты могут иметь разную длину (один обрывается раньше другого), то возможна ситуация, когда точке, лежащей на внешнем мосту, не соответствует никакая точка с внутреннего, поскольку к данному моменту тот уже оборвался. В этом случае происходит соединение этой точки с внешнего моста с последней точкой внутреннего моста.

Таким образом, после формирования поверхности у нас имеется набор примитивов (маленьких треугольников и четырехугольников с вершинами в насчитанных точках), описывающий сингулярную поверхность.

2. Параллельная программа

Из приведенного описания счетных алгоритмов видно, что процедура построения набора одного моста является существенно последовательной: последующее сечение не может быть построено и обработано раньше предыдущего. Распараллеливание процедуры построения и обработки очередного сечения кажется нецелесообразным, поскольку отдельные ее шаги (построение выпуклой оболочки, поиск точек переключения и сингулярных точек) также существенно последовательны и, кроме того, достаточно быстры. В то же время построение наборов сечений разных мостов может вестись независимо.

Поэтому при таком распараллеливании этого алгоритма задача, выполняемая на одном процессоре, — это построение одного моста. Наиболее подходящая схема работы — это "процессорная ферма". При этом один из процессоров выделяется и назначается "мастером" или "ведущим", остальные — "рабочие" или "ведомые". В обязанности "рабочего" входит выполнение задания, полученного от "мастера" с дальнейшей выдачей результатов "мастеру" или непосредственно потребителю. Основной задачей "мастера" является синхронизация работы ансамбля процессоров: раздача заданий на счет, прием результатов, управление доступом к разделяемым внешним ресурсам (например в случае, когда все "рабочие" должны записывать какую-либо информацию в один файл на жестком диске) и т.д. Кроме того, "мастер" сам может осуществлять и счет заданий. Именно так было сделано при реализации программы, поскольку администрирование процессорной фермы занимает не слишком много времени и "мастер" вполне может производить счет.

Оказалось, что различия программ для "мастера" и "рабочего" не слишком велики. Эти выполняемые модули различаются лишь наличием (отсутствием) блока, отвечающим за администрирование работы процессорной фермы. Поэтому было решено разрабатывать выполняемый модуль, одинаковый для "мастера" и "рабочего". Введены специальные проверки, подавляющие выполнение функций администрирования на рабочем процессоре. Блок-схема программы приведена на рис. 2.

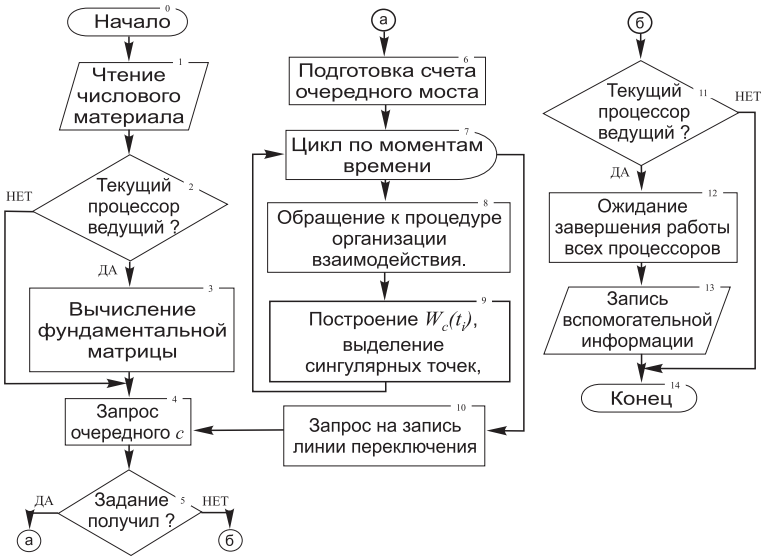

Рис. 2. Блок-схема выполняемого модуля

Следует обратить внимание на блок номер 8 “Обращение к процедуре организации взаимодействия”. Поскольку истинная вытесняющая многозадачность на машине МВС-100 не реализована, то необходимо явно передавать управление процедуре, которая отслеживает запущенные межпроцессорные обмены и инициирует по необходимости новые.

Также необходимо отметить действия, выполняемые в блоке номер 10 “Запрос на запись линий переключения”. Данные по поверхностям переключения сводятся в один файл, однако собираются они с разных мостов, т.е. с разных процессоров. Первоначально задания на счет выдавались по возрастанию параметра c. Это было связано с тем, что множества уровня для меньших значений c имеют, по крайней мере, не большее число сечений, чем множества для бóльших c. Следовательно, множества для меньших c считаются быстрее. Это позволяло “естественно” синхронизировать поступление данных по линиям переключения, и они записывались в выходной файл группами точек, снятыми с одного моста. Но такой подход приводил к

неэффективному распределению счетных заданий между процессорами. В более новых версиях программы задания выдаются, начиная с бóльших c, т.е. начиная с самых длинных заданий. Такое изменение порядка счета привело к рассинхронизации поступления данных по линиям переключения. Это потребовало дополнительного управления записью в выходной файл, которое реализовано в этом блоке.

При разработке программы были встречены следующие проблемы. Во-первых, рассинхронизация межпроцессорных обменов: может произойти, что один из "рабочих" процессоров совершит аварийную остановку (например из-за некорректной арифметической операции или из-за ошибки защиты памяти при выходе за границу массива). Тогда происходит "ступор" всей системы: "мастер" ожидает от этого рабочего каких-либо посылок до тех пор пока вся задача не будет снята с выполнения по истечении заказанного времени счета. Вторая возможность рассинхронизации заключалась в несовпадении длин посылаемого и принимаемого пакетов. В одной из последних версий системных библиотек функции обмена контролируют совпадение этих длин.

Во-вторых, развитие однажды спроектированной системы обменов требовало значительных усилий. В целом программирование межпроцессорных обменов весьма напоминало программирование для системы MS Windows 3.1: обычными были длинные операторы ветвления, которые отслеживали и обрабатывали те или иные посылки.

В процессе работы над параллельной программой (т.е. в течение 1996–1998 гг.) разработаны библиотеки процедур, устраняющие эти недостатки.

2.1. Улучшенные процедуры обмена

Операционной системой предоставляются 6 базовых процедур для организации межпроцессорных обменов (транзакций): запустить прием/передачу данных, проверить окончание приема/передачи, дождаться окончания приема/передачи. Как отмечалось выше, если одним процессором запущен прием, но с

другой стороны не запущена передача (или наоборот), то такая транзакция не будет завершена никогда.

Для устранения такой ошибочной ситуации были написаны надстройки над стандартными процедурами, которые дополнительно могут прерывать работу системы по истечении таймаута транзакции. Это означает следующее. При инициировании посылки (приема) запоминается время начала транзакции. Далее при каждом обращении к какой-либо из процедур обмена проверяются времена, прошедшие с начала каждой транзакции. Если хотя бы одна транзакция была начата ранее, чем указанное время, то транзакция считается неуспешной и производится заданное действие. По умолчанию, таймаут (время ожидания) выставляется равным 2 мин. Пользователь имеет возможность менять это время по своему усмотрению.

Стандартно при превышении времени ожидания пользовательская программа уведомляется возвращением специального кода ошибки. Кроме того, пользователь может заказать в этом случае прекращение работы программы или выполнение своей собственной процедуры.

В этой библиотеке предусмотрена возможность ведения журнала (лога) транзакций, в котором отображается информация по межпроцессорным обменам: номер процессора-инициатора, номер процессора-получателя, время начала, длина посылаемого (принимаемого) пакета и т.д. Такая информация может оказаться полезной при отладке программ.

2.2. Построение системы простейшей кооперативной многозадачности

Одновременно с разработкой параллельной версии происходила модификация и последовательной программы. Причем некоторые изменения были настолько кардинальными, что требовали существенной переработки параллельного алгоритма и как следствие системы межпроцессорных взаимодействий. Необходимость облегчить разработку и модификацию системы обменов привела к написанию следующей библиотеки.

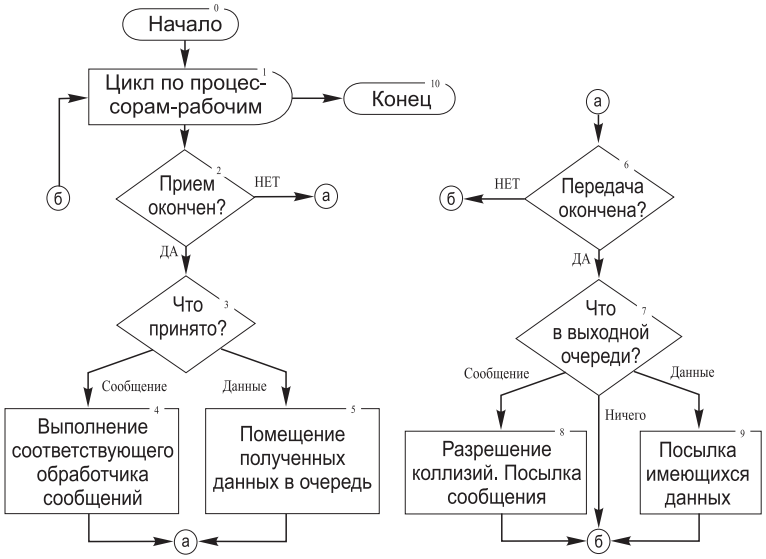

Рис. 3. Блок-схема менеджера обменов

В целом при построении межпроцессорных обменов использована идеология кооперативной многозадачности, которая реализована, например, в операционной системе MS Windows 3.1. Общая схема построения и исполнения программ выглядит при этом следующим образом. Программа пользователя строится не как единый модуль, а как набор независимых процедур, использующих общие данные. Головной программой при этом является не модуль пользователя, а некая системная процедура — *ядро системы*. При изменении состояния системы (например при завершении обмена) инициируется *событие* — внутреннее оповещение ядра. Возникновение события приводит к вызову ядром системной или пользовательской процедуры, связанной с данным событием. Эта процедура называется *обработчиком сообщения*. Управление возвращается ядру при окончании работы обработчика сообщения.

Отсюда название “кооперативная многозадачность”: каждый обработчик сообщения может удерживать управление системой столь долго, насколько это необходимо. Но при этом система на-

ходится в его монопольном распоряжении и все прочие процессы прекращают выполнение.

Следует отметить, что проектирование и написание программ, ориентированных на выполнение в системе, управляемой сообщениями, отличается от создания программ классической архитектуры. Поэтому пользователь должен разработать системы сообщений, процедуры обработки этих сообщений и продумать взаимодействие обработчиков.

Центральной процедурой при такой организации программы является процедура менеджера обменов. Общая блок-схема менеджера обменов приведена на рис. 3. Как отмечалось выше, в данном случае эта процедура является ядром, реализующим кооперативную многозадачность. В обязанности менеджера обменов входит:

1) запуск и своевременное поддержание обменов данными между процессорами;
2) запуск процедуры пользователя, начинающей счет задачи;
3) синхронизация доступа к разделяемым данным — счетные задания, общие файлы на жестком диске головной машины и т.д.
4) своевременная обработка сообщений, генерируемых как завершением обменов между процессорами, так и пользовательскими процедурами;
5) корректная остановка системы в случае исключительных ситуаций (аварийные остановы отдельных процессоров, превышение таймаутов транзакций, нарушение протокола обмена между процессорами, нарушение доступа к разделяемым данным и т.д.).

Следует отметить, что менеджер обменов не является процедурой, исполняемой на каком-то одном процессоре. На каждом процессоре исполняется копия менеджера обменов. Управление системой строится на взаимодействии менеджеров обменов, исполняемых на разных процессорах.

Менеджер обменов имеет возможность программного управления режимом слежения за состоянием системы. Пользователь

из своих процедур может по своему усмотрению включать и выключать протоколирование обменов сообщениями и данными.

К недостаткам такой реализации межпроцессорных обменов можно отнести некоторое снижение эффективности распараллеливания, возникающее вследствие увеличения доли накладных расходов при обмене данными. Увеличение накладных расходов возникает из-за того, что для проверки корректности функционирования системы требуются дополнительные, не запланированные пользователям обмены между процессорами. Однако в случае, когда вычислитель используется для проектирования и отработки новых алгоритмов, дополнительная надежность системы, гарантируемая таким подходом к организации взаимодействия процессоров, полезна и ускоряет разработку и отладку программ.

3. Пример

В качестве демонстрационного примера была выбрана игра "конфликтно-управляемый осциллятор". Эта игра описывается следующей динамикой:

$$\begin{aligned}
&\dot{x}_1 = x_2 + v,\\
&\dot{x}_2 = -x_1 + u,\\
&t \in [0;\, 8], \quad |u| \le 1,\ |v| \le 0.9,\\
&\varphi(x_1, x_2) = x_1^2 + x_2^2.
\end{aligned}$$

На рис. 4 приведены три множества уровня для значений $c = 1.05;\ 1.4;\ 2.7$. Изображение дано в исходных координатах x_1, x_2. Обратное время τ идет слева направо. Для рассмотрения внутренней структуры этих множеств они рассечены плоскостью.

На рис. 5 изображены два вида сингулярных поверхностей, возникающих в этой игре. Следует отметить, что это типичное строение сингулярных поверхностей для данного класса игр. Около оси времени располагается рассеивающая поверхность обоих игроков, которая при удалении от оси времени переходит в

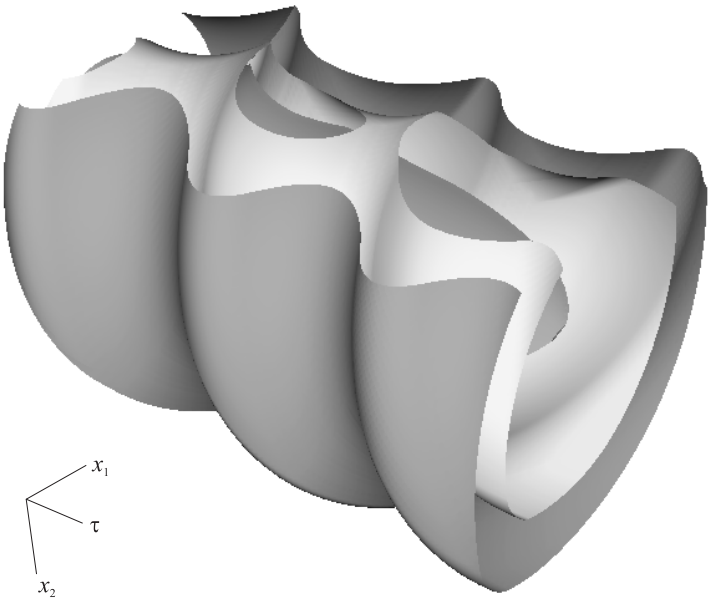

Рис. 4. Три множества уровня для игры “конфликтно-управляемый осциллятор”

экивокальную поверхность. При дальнейшем отходе от оси времени экивокальная поверхность расщепляется на поверхность переключения первого игрока и рассеивающую поверхность второго игрока.

Список литературы

1. *Красовский Н.Н., Субботин А.И.*, Позиционные дифференциальные игры, М.: Наука, 1974. 455 с.
2. Алгоритмы и программы решения линейных дифференциальных игр. Под. ред. А.И.Субботина и В.С.Пацко. Свердловск, 1984. 295 с.
3. *Тарасьев А.М., Ушаков В.Н.*, Алгоритм построения стабильного моста в линейной дифференциальной игре сближения

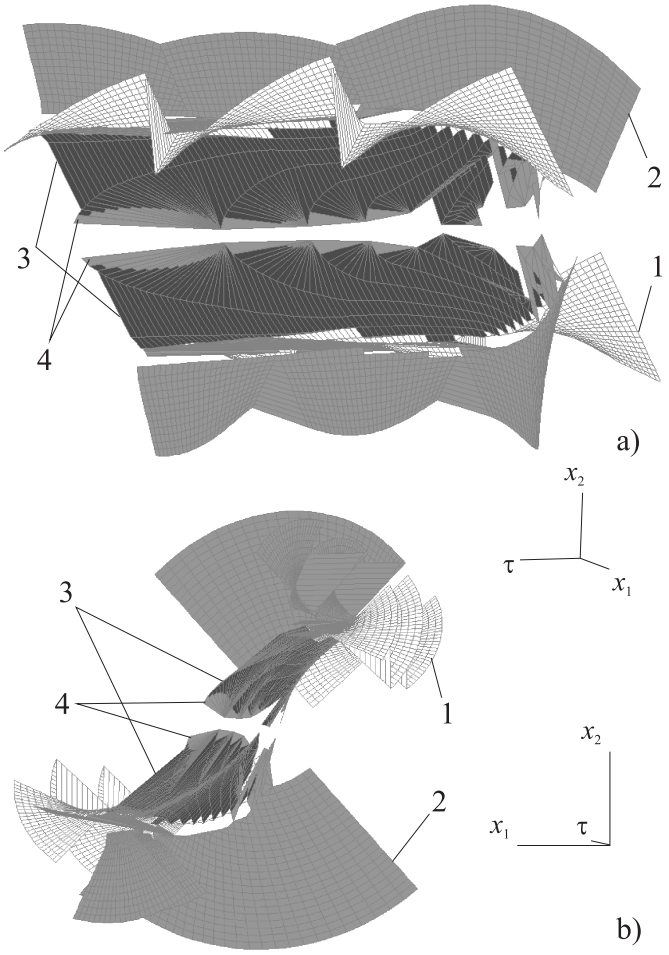

Рис. 5. Сингулярные поверхности для игры “конфликтно-управляемый осциллятор”.
1 — поверхность переключения первого игрока, 2 — поверхность рассеивания второго игрока, 3 — экивокальная поверхность, 4 — поверхность рассеивания обоих игроков

с выпуклым целевым множеством // Исследования минимаксных проблем управления. Свердловск, 1985, С. 82 – 90.
4. *Исакова Е.А., Логунова Г.В., Пацко В.С.*, Построение стабильных мостов в линейной дифференциальной игре с фиксированным моментом окончания // Алгоритмы и программы решения линейных дифференциальных игр. Свердловск, 1984. С. 127 – 158.
5. *Пшеничный Б.Н., Сагайдак М.И.*, О дифференциальных играх с фиксированным временем // Кибернетика. 1970. N. 2. С. 54 – 63.
6. *Боткин Н.Д., Пацко В.С.* Универсальная стратегия в дифференциальной игре с фиксированным временем окончания // Пробл. управления и теории информ. 1982. т.11, №6. С. 419 – 432.
7. *Айзекс Р.*, Дифференциальные игры, М.: Мир, 1965. 480 с.
8. *Melikyan A.A.*, Generalized Characteristics of First Order PDEs: Application in Optimal Control and Differential Games. Birkhauser, Boston, 1998.
9. *Bernhard P.*, Singular surfaces in differential games // Lecture Notes in Control and Information Sciences. Ed. by P. Hagedorn, H.W. Knobloch, G.J. Olsder. Springer-Verlag, Berlin, 1977. V. 3. P. 1 – 33.
10. *Merz A.W.*, The homicidal chauffeur – a differential game, PhD dissertation. Standford University, 1971.
11. *Shinar J., Zarkh M.*, Pursuit of a faster evader – a linear game with elliptical vectograms // Proc. Seventh International Symposium on Dynamic Games. Yokosuka, Japan, 1996. P. 855 – 868.