УДК 519.7, 519.17

РАЗРАБОТКА И ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНЫХ КОМБИНАТОРНЫХ АЛГОРИТМОВ

Н. Е. Тимошевская

Томский государственный университет, г. Томск, Россия

E-mail: timnat@isc.tsu.ru

В русле современного развития прикладной дискретной математики лежит проблема создания эффективных параллельных алгоритмов решения комбинаторных задач. В данной работе излагаются результаты, полученные автором в разное время в этом направлении: 1) методы распараллеливания комбинаторных алгоритмов — методы назначаемых и выделяемых поддеревьев для параллельного обхода дерева поиска в глубину и метод нумерации для параллельного перечисления комбинаторных объектов; 2) параллельные алгоритмы решения комбинаторных задач, разработанные на основе методов распараллеливания, включая задачи перечисления (сочетаний, перестановок, разбиений) и задачи поиска (кратчайшего линеаризационного множества покрытия и решения нелинейной системы логических уравнений методом линеаризационного множества), с экспериментальными оценками их эффективности.

Ключевые слова: *комбинаторные задачи, параллельные алгоритмы, дерево поиска, многопроцессорная система, линеаризационное множество, система логических уравнений.*

Введение

Под комбинаторными задачами в работе подразумеваются *перечислительные* и *поисковые* задачи на конечных комбинаторных множествах, элементами которых служат комбинаторные объекты, представляющие собой комбинации элементов других конечных (возможно, тоже комбинаторных) множеств — сочетания, перестановки, разбиения, покрытия, линеаризационные множества переменных, решения систем логических уравнений и т. п. Перечислительная задача заключается в перечислении всех элементов некоторого подмножества конечного комбинаторного множества, а поисковая — в нахождении в последнем элемента с заданными свойствами. За редким исключением перебор (порождение, перечисление) элементов основного комбинаторного множества и их опробование (проверка на обладание нужным свойством) является единственным способом решения таких задач. Перечисляемые элементы порождаются обычно в некотором порядке. Во многих случаях это делается в порядке обхода дерева поиска, представляющего собой модель комбинаторного множества.

Почти все комбинаторные алгоритмы имеют экспоненциальную сложность. Вместе с тем на практике необходимо уметь решать комбинаторные задачи достаточно больших размерностей. Одна из возможностей уменьшения времени решения комбинаторных задач состоит в том, чтобы разбить комбинаторное множество на классы и исследовать эти классы параллельно, т. е. одновременно на нескольких процессорах вычислительной системы. Проблема заключается в том, как надо разбивать комбинаторное множество на классы и раздавать последние процессорам так, чтобы обеспечить наивысшую степень ускорения вычислений. Последнее зависит от многих факторов,

в том числе от неравномерной загрузки процессоров системы, от накладных расходов на коммуникации (что особенно важно для систем с распределенной памятью), от избыточности просматриваемой области поиска и др. В работе [1] представлен обзор существующих методов разработки параллельных алгоритмов решения комбинаторных задач.

1. Методы и алгоритмы параллельного обхода дерева

Во многих методах решения комбинаторных задач дерево используется в качестве модели пространства возможных решений, а его обход — как модель алгоритма поиска решения. Применение многопроцессорной вычислительной системы позволяет обход дерева *распараллелить* — воспроизводить одновременно несколько узлов в дереве. В [2] предложены два метода параллельного обхода дерева — *метод назначаемых поддеревьев* (МНП) и *метод выделяемых поддеревьев* (МВП). В первом упор делается на сокращение взаимодействия между процессорами системы, во втором — на их равномерную загрузку. В обоих методах параллельные процессы подразделяются на управляющий и рабочие. Под *процессом* понимается вычислительная ветвь программы, выполняющаяся на отдельном процессоре.

1.1. Метод назначаемых поддеревьев

Идея МНП заключается в разбиении дерева на большое число «маленьких» поддеревьев и назначении их для обхода в порядке освобождения процессов. На первом этапе управляющий процесс выполняет построение и обход части дерева в ширину, начиная от его корня до тех пор, пока число построенных, но еще не обработанных вершин (называемых *корневыми*) не достигнет заранее заданного числа m, которое много больше числа рабочих процессов. На втором этапе корневые вершины посылаются рабочим процессам по мере выполнения теми обходов в глубину ранее назначенных им поддеревьев, которые могут быть разного размера.

В данном методе управление работой процессов не требует значительных затрат. Рабочие процессы обмениваются данными с управляющим процессом лишь при назначении очередного поддерева. Вместе с тем метод не гарантирует равномерной загруженности всех процессоров, так как существует опасность, что некоторые процессы получат поддеревья, во много раз превышающие другие по размеру. Для предупреждения этого можно увеличить число m назначаемых поддеревьев. С другой стороны, это приведет к увеличению числа обменов между управляющим и рабочими процессами. Таким образом, МНП рекомендуется для использования в тех случаях, когда можно в результате разбиения получить поддеревья, близкие по размеру.

1.2. Метод выделяемых поддеревьев

В методе выделяемых поддеревьев разбиение на подзадачи происходит не сразу, но по мере выполнения обхода. На первом этапе управляющий процесс выполняет обход дерева в ширину до тех пор, пока число корневых вершин не достигнет числа s рабочих процессов. На втором этапе корневые вершины распределяются по процессам, и каждый процесс начинает выполнять обход в глубину соответствующего поддерева. По окончании обхода процесс освобождается. Для его загрузки выбирается поддерево, обход которого в этот момент ведется одним из рабочих процессов, и в нем выделяется поддерево, которое передается для обхода освободившемуся процессу. При выборе поддерева проверяется некоторое *условие разбиения*, показывающее целесообразность дробления этого поддерева. Например, максимальный номер полностью пройденного яруса должен быть много меньше числа всех ярусов в дереве.

Выбор процесса-источника для выделения поддерева выполняется управляющим процессом по стратегии *оптимального выбора*, нацеленной на то, чтобы направлять запрос к наиболее загруженному процессу. Каждый процесс периодически посылает управляющему информацию о пройденной части поддерева. Управляющий на основании собранной им информации выбирает из рабочих процессов, для которых еще не нарушено условие разбиения, тот процесс, который имеет наибольшую непройденную часть поддерева, и пересылает ему номер освободившегося процесса. Получив указанный номер, процесс-источник выделяет в своем дереве поддерево для свободного процесса. Для выделения поддерева по возможности большего размера ищется непройденная вершина, минимально удаленная от корня, которая и становится корневой вершиной выделяемого поддерева. Таким образом, этот метод позволяет обеспечить равномерную загруженность процессов системы, однако требует дополнительных накладных расходов на выделение поддеревьев.

1.3. Экспериментальное исследование методов параллельного обхода

Предложенные методы пригодны для обхода дерева любого типа. Их экспериментальное исследование проводилось как на полных k-ичных деревьях, так и на деревьях, возникающих при решении некоторых известных комбинаторных задач — перечислительных (перечисление сочетаний и разбиений множества), поисковых (поиск подмножеств с заданной суммой — решений задачи о рюкзаке, поиск решений системы нелинейных логических уравнений) и оптимизационных (поиск оптимального назначения, поиск кратчайшего линеаризационного множества покрытия). Целью эксперимента являлась оценка величин ускорения и эффективности. На практике *ускорение* определяется отношением времени решения задачи на одном процессоре ко времени решения той же задачи на системе таких же процессоров, а *эффективность* — отношением ускорения к числу процессоров. Эффективность показывает среднюю долю времени выполнения программы, в течение которого процессоры реально используются для решения задачи, и не превышает 1. Результаты исследования (см. [2, 3]) показали высокую эффективность (больше 0,8) обоих методов для почти всех типов деревьев. Только в случае «глубоких» деревьев сочетаний, поддеревья которых сильно различаются по числу вершин, результаты эксперимента подтвердили, что МНП работает плохо. В то же время результаты экспериментального анализа показали, что для задачи о назначении МНП имеет преимущество. Эта задача относится к задачам оптимизации, для которых, так же как и для задач поиска, в которых требуется найти хотя бы одно решение, имеет место *эффект несовпадения просматриваемых областей* поиска для параллельного и последовательного алгоритмов и, как следствие, выполнение ими работы разного объема. В параллельном случае он может оказаться как меньше, что приводит к суперлинейному ускорению, так и больше (избыточность просматриваемой области поиска), чем в последовательном. При проведении экспериментального анализа указанный эффект наблюдался при решении задач о назначении и поиске решения системы нелинейных логических уравнений.

2. Параллельное перечисление комбинаторных объектов методом нумерации

Метод нумерации дает альтернативный обходу дерева способ параллельного перечисления элементов комбинаторного множества. Алгоритмы перечисления строятся обычно по следующим правилам: на множестве перечисляемых объектов вводится некоторый порядок, описывается способ преобразования текущего объекта в следую-

щий за ним и формулируется условие окончания перечисления. Для параллельного перечисления множество объектов надо предварительно разбить на классы, мощности которых отвечают производительности процессоров в системе, и перечислять объекты в различных классах независимо и параллельно на соответствующих процессорах. Однако, когда речь идет о множестве комбинаторных объектов, в отличие, скажем, от числового множества, провести такое разбиение в явном виде затруднительно. Решение этой проблемы возможно путем сопоставления перечисляемым комбинаторным объектам элементов некоторого «легко разбиваемого» множества. Самый простой способ состоит в нумерации объектов числами 1, 2, ... в соответствии с установленным на множестве объектов порядком. Последний, естественно, должен допускать восстановление объекта по его номеру, причем время, требуемое на восстановление, должно быть пренебрежимо мало по сравнению со временем, затрачиваемым на перечисление всех объектов одного класса. Предложенный метод нумерации включает метод построения объекта по его номеру в заданном порядке.

Пусть объекты задаются векторами конечной длины вида $(a_1, a_2, \ldots, a_n)$ с компонентами из конечных упорядоченных множеств $A_1, A_2, \ldots, A_n$ $(a_i \in A_i)$. Такой способ представления существует для всех комбинаторных объектов. Будем считать, что объекты упорядочены в соответствии с лексикографическим порядком представляющих их векторов, и их нумерация начинается с 1. *Префиксом* объекта (представляющего его вектора) $(a_1, \ldots, a_n)$ называют вектор $(a_1, \ldots, a_t)$, $0 \leqslant t \leqslant n$. Если $t = 0$, то имеем пустой префикс ().

Пусть $p = (a_1, \ldots, a_{t-1})$ есть некоторый префикс. Обозначим $N(p)$ число объектов, которые имеют префикс p. Множество претендентов на следующую позицию за p, т. е. множество значений, которые может принимать элемент a_t, обозначим $Z(p)$. Следующие теоремы дают алгоритмы вычисления номера объекта и построения объекта по его номеру.

Теорема 1. Пусть комбинаторные объекты упорядочены в соответствии с лексикографическим порядком представляющих их векторов и пронумерованы числами натурального ряда. Тогда номер I объекта $(a_1, a_2, \ldots, a_n)$ вычисляется по формуле

$$I = \sum_{t=1}^{n-1} \sum_{\substack{x \in Z(a_1, a_2, \ldots, a_{t-1}), \\ x < a_t}} N(a_1, a_2, \ldots, a_{t-1}, x) + \sum_{\substack{x \in Z(a_1, a_2, \ldots, a_{n-1}), \\ x \leqslant a_n}} N(a_1, a_2, \ldots, a_{n-1}, x).$$

Теорема 2. Пусть $p = (a_1, a_2, \ldots, a_{t-1})$ есть префикс объекта с номером I в лексикографическом ряду, $Z(p) = \{z_1, z_2, \ldots, z_m\}$ и N есть номер первого объекта с префиксом p. Тогда $a_t = z_k$, где k такое, что

$$N + \sum_{j=1}^{k-1} N(a_1, \ldots, a_{t-1}, z_j) \leqslant I < N + \sum_{j=1}^{k} N(a_1, \ldots, a_{t-1}, z_j).$$

Для применения метода к конкретным комбинаторным объектам требуется уметь вычислять число $N(p)$ и множество $Z(p)$. В работах [4, 5] показывается применение метода нумерации для параллельного перечисления сочетаний, перестановок и разбиений множества.

Использование метода нумерации при распараллеливании обеспечивает высокую эффективность благодаря пропорциональному распределению вычислений и отсутствию обменов данными между процессорами, о чем свидетельствуют и результаты

вычислительных экспериментов. Для параллельного перечисления сочетаний эффективность в среднем равна $0,87$, перестановок — $0,94$, эффективность параллельного алгоритма перечисления разбиений множества лежит в диапазоне $0,8 - 0,85$.

3. Параллельное решение системы логических уравнений методом линеаризационного множества

В работах [6–8] представлены результаты исследования, связанные с решением методом линеаризационного множества системы S логических уравнений вида

$$s_t = f_t(x_1, \ldots, x_n), t = 1, \ldots, m,$$

где $x_1, \ldots, x_n$ — булевы переменные; s_t — булевы константы; $f_t(x_1, \ldots, x_n)$ — булевы функции, представленные суммой по модулю 2 элементарных конъюнкций, $t = 1, \ldots, m$. Множество переменных $L \subseteq X = \{x_1, \ldots, x_n\}$ называется *линеаризационным для* S, если при любой фиксации значений переменных в L система S становится линейной. Суть метода состоит в построении некоторого линеаризационного множества и в нахождении такого набора значений переменных в этом множестве, при подстановке которого в систему последняя становится совместной. Поиск набора выполняется алгоритмом, реализующим сокращенный обход бинарного дерева поиска с возвращением. Глубина такого дерева не превосходит мощности линеаризационного множества, и чем она меньше, тем быстрее ищется решение. Предложен параллельный алгоритм обхода дерева для решения системы по методу выделяемых поддеревьев.

Для проведения экспериментального анализа алгоритма использовались как случайным образом сгенерированные системы логических уравнений, так и системы, описывающие поведение генератора ключевого потока Геффе. В первой серии экспериментов решалась задача поиска всех решений системы, требующая обхода всего дерева. Для систем с достаточно большим линеаризационным множеством (свыше 30 переменных) для 12 процессов средняя эффективность равна 0,84. Во второй серии экспериментов параллельный поиск прекращался, как только один из процессов находил решение. Результаты показали, что в этом случае применение параллельного обхода дерева может сокращать время поиска в число раз, которое может быть как больше, так и меньше числа процессов. Причиной этого является несовпадение областей поиска при последовательном и параллельном обходах.

В связи с использованием метода линеаризационного множества для решения систем логических уравнений возникают задачи: существования линеаризационного множества заданной мощности для заданной системы S, нахождения кратчайшего линеаризационного множества для S, построения систем уравнений с ограничениями на линеаризационные множества и др. В общем случае эти задачи еще не нашли эффективного решения. В этом направлении получены следующие результаты.

Пусть B есть покрытие множества X. Подмножество $L \subseteq X$ называется *линеаризационным множеством покрытия* B, если $|Y - L| \leqslant 1$ для всех $Y \in B$. Если вернуться к системе логических уравнений S, то здесь X есть множество переменных системы, а B — множество его подмножеств, соответствующих конъюнкциям системы S, каждое из которых состоит из переменных, входящих в соответствующую ему конъюнкцию (с инверсией или без). В этом случае линеаризационное множество покрытия B будет линеаризационным множеством переменных системы S, хотя обратное может и не выполняться. В задачах построения линеаризационных множеств достаточно рассматривать только покрытия, не содержащие одноэлементных блоков. Обозначим $M(X)$ множество всех таких покрытий множества X.

Алгоритм А1 покрытию $B \in M(X)$ в соответствие ставит граф $G(V, E)$, такой, что $V = X$ и $\{u, v\} \in E$, если и только если существует $Y \in B$, что $u \in Y$, $v \in Y$ и $u \neq v$. Построенный так граф G называется *графическим представлением покрытия* B и обозначается $\mathrm{A1}(B)$. Алгоритм А1 имеет полиномиальную сложность.

Утверждение 1. Множество L является линеаризационным для $B \in M(X)$ тогда и только тогда, когда оно является вершинным покрытием графа $\mathrm{A1}(B)$.

Покрытия B и C из $M(X)$ *эквивалентны*, если любое подмножество $L \subseteq X$ является линеаризационным для B тогда и только тогда, когда оно линеаризационное для C. Класс эквивалентности на $M(X)$, содержащий множество $B \in M(X)$, обозначим $[B]$. Следующая теорема дает конструктивный тест эквивалентности двух покрытий.

Теорема 3. Пусть $B, B' \in M(X)$. Покрытия B и B' эквивалентны тогда и только тогда, когда графы $\mathrm{A1}(B)$ и $\mathrm{A1}(B')$ совпадают.

Число блоков в покрытии будем называть его *длиной*. Число $\sigma(B) = |B_0| + \ldots + |B_{k-1}|$ назовем весом покрытия $B = \{B_0, \ldots, B_{k-1}\}$. При поиске некоторого линеаризационного множества заданного покрытия во многих случаях имеет смысл выполнить переход к эквивалентному покрытию, например, меньшей длины или меньшего веса. Покрытие $B \in M(X)$ называется *кратчайшим* или *минимальным*, если соответственно $|B| \leqslant |A|$ или $\sigma(B) \leqslant \sigma(A)$ для любого $A \in [B]$. Покрытие $B = \{B_0, \ldots, B_{k-1}\} \in M(X)$ называется *безызбыточным*, если не существуют $B_i \in B$ и $u \in B_i$, что $B' = \{B_0, \ldots, B_{i-1}, B_i - \{u\}, B_{i+1}, \ldots, B_{k-1}\} \in M(X)$ и $B' \in [B]$. В теоремах 4 и 5 установлены необходимое и достаточное условия безызбыточности покрытия и необходимое условие для кратчайшего покрытия.

Теорема 4. Если покрытие $B = \{B_0, \ldots, B_{k-1}\}$ является кратчайшим, то не существует таких множеств $B_{i_1}, B_{i_2}, \ldots, B_{i_r} \in B$, что $r > 1$, $i_1 < i_2 < \cdots < i_r$ и множество $Y = \bigcup\limits_{j=1}^{r} B_{i_j}$ образует полный подграф в $\mathrm{A1}(B)$.

Теорема 5. Покрытие $B = \{B_0, \ldots, B_{k-1}\} \in M(X)$ является безызбыточным тогда и только тогда, когда для любых $B_i \in B$ и $u \in B_i$ имеет место

$$\exists v \in B_i (u \neq v \& \forall B_j \in B (i \neq j \Rightarrow \neg(\{u, v\} \subseteq B_j))).$$

Доказательство достаточности в теореме 5 содержит **алгоритм А3** преобразования заданного покрытия в эквивалентное безызбыточное.

Для $G(V, E) = \mathrm{A1}(B)$ показывается, что 1) $E \in [B]$ и E безызбыточное, 2) если в $G(V, E)$ нет клики, содержащей более двух вершин, то $|[B]| = 1$. Следующее утверждение позволяет выявить структуру классов эквивалентности покрытий.

Утверждение 2.

1) Для любого множества X, $|X| \geqslant 3$, существует безызбыточное, но не кратчайшее и не минимальное покрытие $B \in M(X)$.
2) Для любого множества X, $|X| \geqslant 5$, существует кратчайшее, но не безызбыточное, а также кратчайшее, но не минимальное покрытие $B \in M(X)$.
3) Любое минимальное покрытие $B \in M(X)$ является безызбыточным для любого X.
4) Для любого множества $X, |X| \geqslant 8$, существует минимальное, но не кратчайшее покрытие $B \in M(X)$.

Пусть $G(V, E)$ есть неориентированный граф без петель и кратных ребер. Покрытие B множества $X = V$, такое, что множество L является линеаризационным для этого покрытия тогда и только тогда, когда оно является вершинным покрытием графа

$G(V, E)$, будем называть *соответствующим графу* G. В работе [7] дается полиномиальный **алгоритм А2** построения соответствующего графу G покрытия, обозначаемого $\mathrm{A2}(G)$. Некоторые его свойства отражены в утверждении 3.

Утверждение 3. Пусть $B = \{B_0, \ldots, B_{k-1}\} = \mathrm{A2}(G(V, E))$. Тогда

1) $k \leqslant |E|$, причем $k = |E|$ тогда и только тогда, когда $|B_i| = 2$ для каждого $i = 0, \ldots, k-1$; в этом случае $B = E$;
2) $|B| < |E|$, если и только если в G есть полный подграф с тремя (или более) вершинами;
3) $\sigma(B) \leqslant |E| + k$, причем $\sigma(B) = 2|E|$ тогда и только тогда, когда $B = E$.

Теорема 6. Задача построения кратчайшего покрытия, эквивалентного данному покрытию, является NP-трудной.

Экспоненциальная сложность алгоритмов минимизации покрытия вынуждает отказаться от поиска точного решения — кратчайшего или минимального покрытия и ограничиться существующим или с меньшими длиной и весом. Последнего в некоторых случаях можно достичь с помощью следующего полиномиального **алгоритма А4**. Последовательно применяются алгоритмы А1, А2 и А3, т. е. вычисляются $G(V, E) = \mathrm{A1}(B)$, $B' = \mathrm{A2}(G)$, $B'' = \mathrm{A3}(B')$ и B'' принимается за результат минимизации B, если $|B''| < |B|$ и $\sigma(B'') < \sigma(B)$.

Теорема 7. Задача построения кратчайшего линеаризационного множества покрытия является NP-трудной.

Следующее утверждение позволяет, во-первых, оценить снизу мощность линеаризационных множеств для покрытий некоторых типов и, во-вторых, разрабатывать алгоритмы построения покрытий с линеаризационными множествами ограниченной снизу мощности.

Утверждение 4.

1) Если в графе $\mathrm{A1}(B)$ есть клика мощности k, то любое линеаризационное множество покрытия B имеет мощность не меньше $k - 1$.
2) Если $\mathrm{A1}(B)$ является полным двудольным графом $K_{t,m}$, то всякое кратчайшее линеаризационное множество для B имеет мощность $\min(t, m)$.
3) Если граф $\mathrm{A1}(B)$ состоит из r компонент связности $G_1, \ldots, G_r$ и для каждого $i = 1, \ldots, r$ в G_i есть клика мощности k_i, то для покрытия B не существует линеаризационного множества мощности меньше $\sum_{i=1}^{r} k_i - r$.

Предложен алгоритм параллельного обхода дерева поиска кратчайшего линеаризационного множества покрытия, разработанный по методу назначаемых поддеревьев. На первом этапе управляющий процесс с помощью жадного алгоритма находит некоторое линеаризационное множество, затем выполняет обход дерева в ширину, начиная от его корня, до тех пор, пока не построит заданное число m корневых вершин. При этом для каждой вершины вычисляется функция нижней оценки и, если она показывает перспективность ветвления из данной вершины, то вершина помещается в очередь. Таким образом, сокращение обхода выполняется уже на первом этапе. На втором этапе управляющий процесс, как и предписывает метод, назначает корневые вершины рабочим процессам по мере обхода ими ранее назначенных поддеревьев. Кроме корневой вершины, управляющий процесс передает рабочему минимальную известную ему мощность линеаризационного множества. В свою очередь, если рабочий процесс находит

в своем поддереве линеаризационное множество меньшей мощности, то пересылает его мощность управляющему. Эффективность распараллеливания достигает в среднем величины 0,9. В компьютерном эксперименте показано также высокое качество жадного алгоритма: в подавляющем числе примеров мощность кратчайшего линеаризационного множества отличается от мощности множества, найденного жадным алгоритмом, не более чем на 2. Кроме того, показана полезность предварительного преобразования заданного покрытия алгоритмом A4, а именно: оно позволяет в среднем сократить число блоков в покрытии в 15 раз, а время поиска кратчайшего линеаризационного множества в некоторых случаях — в сотни раз.

Наконец, построены оценки количества покрытий, имеющих линеаризационные множества заданной мощности. Обозначим $L(n,k)$ число всех покрытий множества $X=\{1,2,\ldots,n\}$ с линеаризационными множествами мощности k.

Теорема 8. $L(n,1)=1+2n(3^{n-1}-n)$,

$$\sum_{s=1}^{2^k-1} D_{ks}(2^{n-k+1}-1)^s \leqslant L(n,k) \leqslant \binom{n}{k} 2^{n-k} \sum_{s=1}^{2^k-1} D_{ks}(2^{n-k+1}-1)^s,$$

где D_{ks} есть число покрытий k-элементного множества, состоящих из s блоков.

Из теоремы следует, что доля покрытий с нетривиальными линеаризационными множествами ничтожно мала, но их количество велико и растет с ростом мощностей покрываемого и линеаризационного множеств, оправдывая постановку задачи поиска линеаризационных множеств для них.

ЛИТЕРАТУРА

1. *Тимошевская Н. Е.* О методах разработки параллельных комбинаторных алгоритмов // Третья Сибирская школа-семинар по параллельным вычислениям / Под ред. А. В. Старченко. Томск: Изд-во Том. ун-та, 2006. С. 60–72.
2. *Тимошевская Н. Е.* Параллельные методы обхода дерева // Математическое моделирование. 2004. Т. 16. № 1. С. 105–114.
3. *Беляев В. А., Тимошевская Н. Е.* Распараллеливание обхода дерева поиска для решения задачи о рюкзаке на кластерной системе // Высокопроизводительные параллельные вычисления на кластерных системах: Материалы Междунар. науч.-практич. сем. / Под ред. проф. Р.Г. Стронгина. Н. Новгород: Изд-во Нижегор. ун-та, 2002. С. 16–20.
4. *Тимошевская Н. Е.* О нумерации перестановок и сочетаний для организации параллельных вычислений в задачах проектирования управляющих систем // Изв. Томского политехнического университета. 2004. Т. 307. № 6. С. 18–20.
5. *Тимошевская Н. Е.* Параллельное перечисление разбиений множества методом нумерации // Вестник Томского госуниверситета. Приложение. 2006. № 17. С. 260–264.
6. *Тимошевская Н. Е.* Задача о кратчайшем линеаризационном множестве // Вестник Томского госуниверситета. Приложение. 2005. № 4. С. 79–83.
7. *Тимошевская Н. Е.* О линеаризационно эквивалентных покрытиях // Вестник Томского госуниверситета. Приложение. 2005. № 4. С. 84–91.
8. *Тимошевская Н. Е.* Параллельные вычисления в решении систем логических уравнений методом линеаризации // Материалы XV Междунар. школы-семинара «Синтез и сложность управляющих систем» (Новосибирск, 18–23 октября 2004 г.). Новосибирск: Институт математики, 2004. С. 97–102.