Управление, вычислительная техника и информатика

УДК 681.3.06

ОПРЕДЕЛЕНИЕ ОБЪЕМОВ ПЕРЕДАЧ ДАННЫХ В СЕТИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ ДЛЯ ЗАДАННОЙ МОДЕЛИ ПРОГРАММНОЙ НАГРУЗКИ

А.В. Погребной

Институт «Кибернетический центр» ТПУ
E-mail: Sasha@ad.cctpu.edu.ru

Выделены факторы, которые определяют объем данных, передаваемых между станциями многопроцессорной вычислительной системы. Предложены методики построения информационного графа модели программной нагрузки и его разрезания для формирования плана использования ресурсов станций. Показано, что критерий, принятый в задаче разрезания, соответствует минимизации объема передаваемых данных.

Введение

При проектировании распределенных систем реального времени (СРВ) снижение затрат времени на выполнение прикладных функций за счет распараллеливания приводит к росту объемов передач данных в сети вычислительной системы. Если число микропроцессорных станций вычислительной системы задано, то объем передаваемых данных между станциями сети зависит от следующих факторов – условий функционирования объекта управления, значений параметров программных модулей, реализующих прикладные функции СРВ и составляющих основную часть программной нагрузки на систему, распределения программной нагрузки по станциям.

Объем передаваемых по сети данных соответствует определенным затратам времени, которые увеличивают времена завершения выполнения прикладных функций программной нагрузки. Затраты времени на передачу данных по сети могут оказаться соизмеримыми с временем работы процессоров при выполнении модулей программной нагрузки. Поэтому при проектировании СРВ анализ влияния перечисленных факторов на объем передаваемых данных в сети, имеет важное значение для принятия решений при проектировании структуры сети вычислительной системы.

Современные СРВ, как правило, разрабатываются в сетевом варианте. Разработчик распределенной СРВ, приступая к выполнению проекта, испытывает большую неопределенность относительно необходимой совокупности станций разных видов, их размещения на территории объекта, топологии сети, способной своевременно передавать данные между станциями, распределения программной нагрузки по станциям. Эти задачи взаимосвязаны, но методов их совместного решения нет. Поэтому автономно решается задача определения числа станций, после этого вручную принимается вариант топологии сети и далее решается задача распределения программной нагрузки по станциям чаще всего по критерию минимальной загрузки сети [1, 2].

Для многих других приложений, не связанных с проектированием жестких СРВ (автоматизация проектных и научных работ, управление производством и т. п.), число станций и топология сети предопределяется объектом и принимается заданной. В этих случаях основной становится задача распределения программной нагрузки по станциям сети и получение плана использования ресурсов. Известно много работ, в которых задача получения плана использования ресурсов формулируется как нелинейная задача математического программирования с булевыми переменными, например [1]. Такие задачи можно решать, используя метод линеаризации, но как отмечено в [1] для распределения 25 программных модулей по 15 станциям получается линейная задача на 2500 переменных и 8000 ограничений.

Применительно к распределенным СРВ обе задачи подробно изложены в [2]. Задача определения числа станций и их размещения в [2] ставится как задача линейного программирования, но уже для территории размещения станций, представленной сеткой размерностью 20×50, число переменных достигает одного миллиона. Задача получения плана использования ресурсов в [2] сформулирована как нелинейная задача математического программирования с булевыми переменными для двух вариантов топологии сети.

Помимо большой размерности задач, основной недостаток таких подходов заключается в том, что при решении задачи получения плана использования ресурсов, топология сети принимается заданной. Если топологию сети учитывать путем введения дополнительных переменных, то получается неприемлемая для решения задача.

Существенным недостатком является также и то, что не учитываются условия функционирования объекта управления, такие как условия поступления входных данных и, соответственно, запуска процессов, условия обновления состояний выходных данных, правила селекции состояний выходных данных при передаче их между процессами. Перечисленные условия непосредственно влияют на объем данных, передаваемых в сети.

Следуя технологии модульного проектирования распределенных СРВ [3], предлагается другой подход к решению рассматриваемых задач, в котором условия функционирования объекта отображаются на модель программной нагрузки и учитываются при определении минимального объема данных, передаваемых между станциями. Задача определения топологии сети решается для полученного объема данных по критерию минимальных затрат времени на передачу данных между станциями сети. Эта задача особенно важна при проектировании жестких СРВ. Для определения числа станций используется метод, изложенный в [4]. При этом все постановки задач имеют размерность, приемлемую для практического применения.

Основная цель исследований, представленных в данной статье, сводится к тому, чтобы для заданных условий функционирования объекта, модели программной нагрузки и числа станций отобразить условия функционирования объекта на модель программной нагрузки и определить объем данных, который потребуется передавать между станциями при функционировании СРВ. При этом задача распределения программной нагрузки по станциям сформулирована как задача разрезания графа на минимально связанные подграфы. В статье также определяется оценка качества вариантов разрезания и вводится граничная величина этой оценки.

Исследования выполнены в условиях, когда модель программной нагрузки представлена в форме графа потока данных (ГПД), реализующего прикладные функции проектируемой СРВ [3]. В ГПД два вида вершин – фрагменты алгоритмов (модули) и переменные (данные). Дуги в графе связывают модули и данные и указывают на отношение модулей к потреблению и формированию тех или иных данных.

Анализ условий потребления ресурсов

При составлении плана использования ресурсов станции рассматриваются как объекты, предоставляющие ресурсы – процессоры и память. Объектами потребления этих ресурсов являются вершины ГПД – данные и модули. Наличие ресурсов и их потребление оценивается на некотором интервале времени названном циклом моделирования. Величина интервала времени цикла моделирования подбирается так, чтобы отрезок времени (цикл обновления [1]), отводимый на однократное выполнение для любой прикладной функции СРВ, укладывался в нем целое число раз. Например, если циклы обновления для прикладных функций СРВ соответствуют отрезкам $\{t_k\}=\{2,3,5,10\}$, то цикл моделирования принимается равным наименьшему общему знаменателю $t^*=30$ либо величине μt^*, $\mu=2,3,....$

Задача составления плана заключается в том, чтобы для всех объектов ГПД (данные, модули, программы) указать конкретные станции, ресурсы которых они будут использовать. При назначении данных на станцию используется ресурс памяти. Модуль в общем случае может назначаться на две станции. Для упрощения задачи будем считать, что операция и программа модуля всегда потребляет ресурсы только одной станции. В этом случае объектами потребления ресурсов становятся модули и данные, что в явном виде и отражает ГПД. Принимается также, что данные, как и модули, не могут делиться на части, то есть назначаются на ресурсы станции целиком. Задача планирования, таким образом, сводится к задаче распределения модулей и данных по станциям.

Для распределения модулей и данных необходимо установить для них объемы потребляемых ресурсов. Ресурс процессора будем оценивать временем, которое отводится на выполнение модулей в одном цикле моделирования. Соответственно для каждого модуля $f_m \in F$, $m=1,2,...,M$, процессорное время, потребляемое модулем за один цикл моделирования, составит величину $\tau_m \rho_m$. Здесь τ_m – время выполнения модуля f_m, а ρ_m – частота выполнения модуля f_m в одном цикле моделирования.

Память i-ой станции, выделяемую для хранения компонентов ГПД, обозначим величиной P_i, $i=1,2,...,n$. В памяти размещаются входные и выходные данные, программы модулей и промежуточные данные, формируемые и потребляемые модулями. Входные и выходные данные $d_q \in D$, $q=1,2,...,Q$, требуют для размещения память p_q и обновляются в соответствии с циклами поступления и обновления. Программы модулей требуют память p_m и занимают ее на всем цикле моделирования.

Промежуточные данные делятся на два вида. К первому виду относятся данные $d_q \in D$, состояния которых обновляются после каждого выполнения модуля производителя и требуют память в размере p_q. Второй вид соответствует ситуации, когда необходимо хранить несколько состояний данного $d_q \in D$, полученных согласно заданному правилу присоединения состояний [3]. Наличие на входе у модуля потребителя данного с некоторым множеством состояний предполагает возможность селекции (выбора) из этого множества определенных состояний. При разработке ГПД правила обновления с присоединением состояний должны быть согласованы с правилами селекции. Чаще всего правила

селекции диктуют необходимость использования определенных правил присоединения. Возможна, также ситуация, когда число присоединяемых состояний превышает число состояний, получаемых модулем за один цикл моделирования. Во всех этих случаях данное $d_q \in D$, для которого определено правило присоединения b_q состояний, требует память в размере $b_q p_q$.

Построение информационного графа для модели программной нагрузки

Объем данных, передаваемых по дугам ГПД (d_q,f_m) и (d_m,f_q), вычисляется для одного цикла моделирования и принимается в качестве весов дуг информационного графа. Для решения задачи распределения модулей и данных не имеет значения в каком направлении передаются данные – к модулю или от него. Поэтому на основе ГПД можно построить информационный граф с матрицей смежности вершин $A=\|a_{qm}\|_{Q\times M}$, $a_{qm}=1$, если данное d_q инцидентно модулю f_m, $a_{qm}=0$, в противном случае.

Вес r_{qm} дуги (d_q,f_m) зависит от размера памяти p_q, занимаемой данным d_q, частоты передачи данного d_q по дуге (d_q,f_m) и числа присоединяемых состояний. Частота передачи данного по дуге (d_q,f_m), инцидентной модулю f_m, равна частоте выполнения этого модуля ρ_m в цикле моделирования. Правила присоединения и селекции определяют число состояний c_{qm} данного d_q, передаваемых по дуге (d_q,f_m).

На основе матрицы A, величин $p_q\rho_m$ чисел передаваемых состояний c_{qm} строится матрица весов $R=\|r_{qm}\|_{Q\times M}$, элементы которой определяются по выражению

$$r_{qm} = a_{qm} c_{qm} p_q \rho_m . \qquad (1)$$

Пример ГПД, содержащего 5 модулей и 9 данных, показан на рис. 1. В ГПД данные d_q пронумерованы в кружках, номера модулей f_m проставлены рядом с планками. Число на дуге обозначает частоту передаваемых по ней данных за один цикл моделирования. Рядом с входными данными указываются условия их поступления, например Ц(2). Здесь в скобках указано число тактов цикла поступления и запуска соответствующего процесса. Числа, расположенные рядом с кружками, указывают на число тактов цикла обновления.

ГПД на рис. 1, содержит 4 процесса [3]. Три из них запускаются циклически по условиям поступления Ц(1), Ц(5) и Ц(2) и соответствующими циклами обновления: 1, 5, 2. Четвертый процесс запускается сигналом B с заданной вероятностью поступления и формирует выходной сигнал в группе $k-1$. Из рис. 1 следует, что необходимость хранения и передачи более одного состояния данных возникает в двух случаях. Модуль f_3 запускается после того как модуль f_1 сработает 5 раз. К этому моменту будет получено 5 состояний данного d_4, которое поступает на вход модуля f_3. Аналогично модуль f_4 запускается после того как получено 2 состояния d_5. Поэтому в матрице C, элементы которой определяют число передаваемых состояний, рис. 2, отражено, что все данные предаются по одному состоянию за исключением двух указанных случаев. На вход модуля f_5 от данного d_4 поступает одно из пяти состояний, что определено правилом селекции.

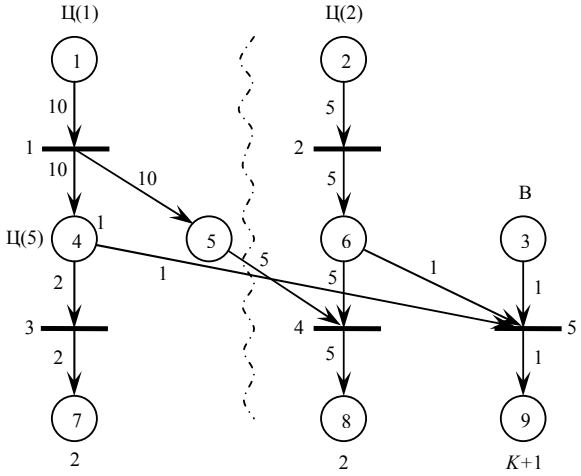

Рис. 1. *Пример ГПД*

$$C=\begin{array}{c|ccccc|c} & 1 & 2 & 3 & 4 & 5 & p_q \\ 1 & 1 & & & & & 2 \\ 2 & & 1 & & & & 3 \\ 3 & & & & & 1 & 1 \\ 4 & 1 & & 5 & & 1 & 2 \\ 5 & 1 & & & 2 & & 4 \\ 6 & & 1 & & 1 & 1 & 2 \\ 7 & & & 1 & & & 1 \\ 8 & & & & 1 & & 1 \\ 9 & & & & & 1 & 1 \\ \hline \rho_m = & 10 & 5 & 2 & 5 & 1 & \end{array} \qquad R=\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 20 & & & & \\ 2 & & 15 & & & \\ 3 & & & & & 1 \\ 4 & 20 & & 20 & & 2 \\ 5 & 40 & & & 40 & \\ 6 & & 10 & & 10 & 2 \\ 7 & & & 2 & & \\ 8 & & & & 2 & \\ 9 & & & & & 1 \end{array}$$

Рис. 2. *Пример построения матриц C и R*

Промежуточное данное d_6 относится к первому виду, то есть обновляется после срабатывания модуля f_2. Справа у матрицы C приведены размеры данных p_q, а снизу ρ_m – частоты выполнения модулей, подсчитанные исходя из цикла моделирования, принятого равным $t^*=10$ тактов. Элементы матрицы R, рис. 2, определены по выражению (1).

Формирование плана использования ресурсов станций

Для решения задачи распределения модулей и данных по ресурсам станций мы имеем информационную графовую форму модели программной нагрузки, для которой определены объемы потребления модулями процессорного времени в размере $\rho_m\tau_m$, памяти для хранения данных в размере $b_q p_q$ и программ модулей в размере p_m, а также объемы данных r_{qm}, передаваемых по дугам графа за один цикл моделирования. Критерием качества решения задачи является нахождение варианта распределения модулей и данных по станциям, доставляющего минимальный суммарный объем данных, передаваемый в локальной сети. Исходя из названного критерия при решении задачи важно,

чтобы модули и данные, между которыми передается большой объем информации, были распределены на одну станцию. Следовательно, решение задачи должно быть таким, чтобы суммарный объем данных, передаваемый по дугам, связывающим модули и данные, размещенные в разных станциях, был минимальным. Такое решение соответствует известной задаче разрезания графа на минимально связанные части [5].

Применительно к рассматриваемой задаче распределения модулей и данных имеем информационный граф для модели программной нагрузки в виде двудольного взвешенного графа $G=(D,F,R)$, где D – множество вершин данных $D=\{d_q\}$ с указанием для каждого $d_q \in D$ размера требуемой памяти $b_q p_q$; F – множество вершин модулей $F=\{f_m\}$ с указанием для каждого $f_m \in F$ величины потребляемого процессорного времени $\rho_m \tau_m$; R – матрица объемов передаваемых данных между вершинами графа.

Сумму весов реберного соединения S_{ij} частей $G_i=(D_i,F_i,S_i)$ и $G_j=(D_j,F_j,S_j)$ графа G обозначим величиной r_{ij}, $r_{ij} = \sum_{s_{qm} \in S_{ij}} r_{qm}$. Здесь s_{qm} – ребро графа G, связывающее вершины d_q и f_m; S_{ij} – множество ребер, связывающих вершины, одна из которых принадлежит части G_i, а вторая – части G_j. Размер частей G_j определяется ресурсами станций по памяти P_i и процессорному времени T_i, которое не может превышать цикл моделирования. В принятых обозначениях задача разрезания графа G на n частей G_i, $i=1,2,...,n$, запишется в виде:

$$\min r = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}; \qquad (2)$$

$$\sum_{d_q \in D_i} b_q p_q \le P_i, \quad i = 1,2,...,n; \qquad (3)$$

$$\sum_{f_m \in F_i} \rho_m \tau_m \le T_i, \quad i = 1,2,...,n. \qquad (4)$$

Разрезание графа G производится на n частей, по числу станций вычислительной системы. При этом должны соблюдаться условия:

$$\sum_{d_q \in D} b_q p_q \le \sum_{i=1}^{n} P_i, \quad \sum_{f_m \in F} \rho_m \tau_m \le \sum_{i=1}^{n} T_i. \qquad (5)$$

Выполнение соотношений (5) обуславливает возможность решения задачи (2)–(4), то есть ресурсов вычислительной системы должно быть достаточно для распределения всех вершин графа с учетом значений параметров по памяти и процессорному времени. Как правило, значения P_i и T_i задаются с некоторым коэффициентом запаса.

Оценка объема данных, передаваемых в сети, для принятого плана использования ресурсов

Наличие матрицы R позволяет оценивать решения задачи разрезания графа G, что соответствует также и оценке решения задачи распределения модулей и данных по станциям. Задачу распределения будем рассматривать как задачу разбиения вершин множества V информационного графа G на подмножества вершин V_i. Вариант разбиения w из множества возможных вариантов W обозначим совокупностью $\{V_i\}_w$ множеств вершин V_i. Варианту $w \in W$ соответствует множество ребер $S_w=S_{ij}$, где S_{ij} – множество ребер, связывающих между собой вершины из множеств V_i и V_j разбиения $\{V_i\}_w$. Таким образом, общий объем передаваемых по сети данных за один цикл моделирования для варианта разбиения $\{V_i\}_w$ составит величину r_w, $r_w = \sum_{s_{qm} \in S_w} r_{qm}$. Получим величину r_w для варианта распределения модулей и данных по двум станциям для примера ГПД (рис. 1). В качестве начальной вершины для формирования множества будем использовать правило выбора вершины d_q с максимальным суммарным весом инцидентных ребер r_q^*, $r_q^* = \max_q \sum_{m=1}^{M} r_{qm}$.

Для нашего примера r_q^* соответствует вершине d_5, то есть сумме весов 5-ой строки матрицы R (рис. 2). Величина $r_5^*=80$ ед. объема. Поэтому вершина d_5 и связанные с ней вершины (модули) f_1 и f_4 включаются в формируемое множество, то есть распределяются на одну станцию. В это же множество включаются вершины d_1,d_6,d_8. При таком распределении модулей и данных множества V_1 и V_2 связаны ребрами $S_w=S_{1,2}=\{(d_4,f_1),(d_6,f_5),(d_6,f_2)\}$. Соответственно, объем передаваемых данных r_w составляет 32 ед. Заметим, что вполне очевидный вариант разбиения, показанный на рис. 1, включает два ребра $S'_w=S'_{1,2}=\{(d_5,f_4),(d_4,f_5)\}$ и дает оценку $r'_w=r'_{1,2}=42$ ед. Эта оценка существенно хуже оценки $r_w=32$ ед.

Для оценки разбиения вершин информационного графа $G=(V,R)$ на подмножества V_j, $j=1,2,...,n$ введем оценку компактности этих подмножеств. Компактность подмножества V_j будем оценивать величиной R_j, равной сумме весов ребер, связывающих вершины этого подмножества. Тогда для оценки компактности варианта разбиения $\{V_j\}_w$ можно использовать величину R_w, $R_w = \sum_{j=1}^{n} R_j$. Оценки R_w и r_w разбиения $\{V_i\}_w$ взаимосвязаны – увеличение оценки R_w соответствует уменьшению оценки r_w.

Вариант разбиения $\{V_j^*\}_w$, который соответствует максимальной оценке R_w^*, будем именовать компактным разбиением (K-разбиением). Для получения разбиения $\{V_j\}_w$ могут быть использованы алгоритмы, предложенные в [4]. Данные алгоритмы позволяют получить разбиения с локальным оптимумом величины R_w. Такие разбиения будем именовать локальными компактными разбиениями (*ЛК*-разбиениями). Поэтому при использовании данных алгоритмов очень важно иметь возможность оценить близость полученного *ЛК*-разбиения к K-разбиению. С этой целью предлагается ввести некоторую граничную оценку компактно-

сти R_0, которую в идеале может получить K-разбиение. Такая оценка вычисляется исходя из предположения о том, что в идеальном разбиении каждая вершина $v_i \in V$, $i=1,2,...,z$ попадает в множество V_j^* с максимально возможной оценкой компактности. В соответствии с этим предположением для каждой вершины $v_i \in V$, сформируем множество V_i, $|V_i|=|V_j^*|$, с максимально возможной оценкой компактности R_i. Тогда граничная оценка компактности R_0 определяется выражением:

$$R_0 = \frac{1}{\mu_j}\sum_{i=1}^{z} R_i, \quad \mu_j = |V_j| = \text{const}.$$

Оценка R_0 является верхней граничной оценкой для оценки R_w^* K-разбиения, то есть между этими оценками должно соблюдаться условие

$$R_w^* \le R_0. \qquad (6)$$

В идеальном случае, когда соблюдается высказанное выше предположение, данные оценки могут совпадать. Это означает, что имеет место идеальное K-разбиение. Оно характеризуется тем, что для вершин, $v_i \in V_j^*$ нельзя построить множества $V_i \ne V_j^*$, $v_i \in V_i$, $|V_i|=|V_j^*|$, с оценкой компактности лучше, чем у множества V_j^*.

Условие (6) нетрудно доказать. Для этого воспользуемся рис. 3. На рис. 3 представлено одно из множеств V_j^* K-разбиения $\{V_j^*\}_w$, содержащее 5 вершин, $\mu_j=5$.

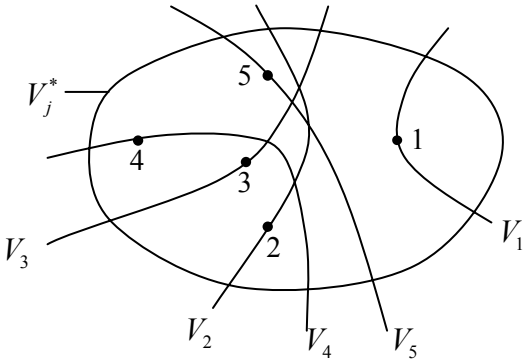

Рис. 3. *Иллюстрация к доказательству условия (6)*

Для каждой вершины построены множества V_i, которые на рис. 3 условно показаны кривыми линиями, проходящими через соответствующие вершины. Оценку компактности множества V_i обозначим величиной R_{ij}. Оценка R_{ij} соответствует наилучшей оценке компактности множества V_i, включающего вершину v_i, то есть $R_{ij} \ge R_j^*$. Это следует из того, что оценка R_{ij} не может быть меньше R_j^*, так как при формировании множества V_i всегда есть возможность сформировать $V_i=V_j^*$, и тогда оценка $R_{ij}=R_j^*$.

Из рис. 3 следует, что все $V_i \ne V_j^*$, поэтому оценки R_{ij} превышают R_j^*. Из этого следует, что усредненная оценка по пяти множествам также превышает оценку R_j^*, то есть $\frac{1}{5}\sum_{i=1}^{5} R_{ij} \ge R_j^*$. Усредненные оценки для других множеств V_j^* K-разбиения $\{V_j^*\}_w$ получим по аналогичному выражению $\frac{1}{\mu_j}\sum_{i=1}^{\mu_j} R_{ij} \ge R_j^*$. Тогда для всех множеств V_j^* можно записать $\sum_{j=1}^{n}\left(\frac{1}{\mu_j}\sum_{i=1}^{\mu_j} R_{ij}\right) \ge \sum_{j=1}^{n} R_j^*$. Учитывая, что $n\mu_j=z$, получаем условие (6), $\frac{1}{\mu_j}\sum_{j=1}^{n}\sum_{i=1}^{\mu_j} R_{ij} = \frac{1}{\mu_j}\sum_{i=1}^{z} R_i = R_0 \ge R_w^*$.

Заключение

Информационный граф, полученный в результате анализа условий функционирования объекта, дает полное представление о передачах данных между вершинами графа в одном цикле моделирования. Это позволяет получать оценки по загрузке сети вычислительной системы в зависимости от планов использования ресурсов, которые определяются вариантами разбиения вершин информационного графа. Наличие таких оценок позволяет принимать решения по корректировке условий функционирования программной нагрузки (условия запуска процессов, обновления и селекции данных) и структуры сети вычислительной системы.

Возможность получения граничной оценки имеет важное познавательное значение при исследовании K-разбиений и практическое – при решении задачи минимизации объемов передач данных. Так, после получения *ЛК*-разбиения, можно оценить его отклонение от граничной оценки и, с некоторой погрешностью, от K-разбиения. Если отклонение велико, то имеет смысл продолжить поиск другого *ЛК*-разбиения с лучшей оценкой. Более того, если оценка этого разбиения совпадает с граничной, то имеет место идеальное K-разбиение.

СПИСОК ЛИТЕРАТУРЫ

1. Chu W.W., Holloway L.J., Lan M.T., Efe K. Task allocation in distributed data processing// IEEE Trans. on Computers. – 1980. – V. 13. – № 11. – P. 57–69.
2. Шенборт И.М., Алиев В.М. Проектирование вычислительных систем распределенных АСУ ТП. – М.: Энергоатомиздат, 1989. – 88 с.
3. Погребной В.К. Системы реального времени. Моделирование и автоматизированное проектирование. – Томск: Изд-во ТПУ, 2006. – 208 с.
4. Погребной А.В. Определение числа и топологии размещения станций многопроцессорной вычислительной системы // Известия Томского политехнического университета. – 2006. – Т. 309. – № 7. – С. 160–164.
5. Пантелеев А.В., Летова Т.А. Методы оптимизации в примерах и задачах. – М.: Высшая школа, 2002. – 544 с.

Поступила 11.12.2006 г.