ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

УДК 681.3.06:519

Н.С. Коваленко[1], В.Н. Венгеров[2], В.М. Метельский[3]

МИНИМИЗАЦИЯ ЧИСЛА ОБРАБАТЫВАЮЩИХ УСТРОЙСТВ ПРИ РАСПРЕДЕЛЕННОЙ ОРГАНИЗАЦИИ ВЫЧИСЛЕНИЙ

Рассматривается математическая модель организации неоднородных распределенных конкурирующих процессов при условии асинхронного режима их взаимодействия для систем с конвейерной организацией распределенных вычислений (СКОРВ). Предлагаются полиномиальные алгоритмы нахождения минимального числа обрабатывающих устройств (ОУ), которые обеспечивают выполнение заданных объемов вычислений за директивное или минимальное время.

Введение

Одной из центральных проблем, возникающих при исследовании и создании математических моделей и методов анализа и организации параллельных процессов, разработке программно-аппаратного обеспечения в СКОРВ, является нахождение условий оптимальной реализации заданных объемов вычислений [1]. Решение этой проблемы приводит к сложным комбинаторно-оптимизационным задачам, связанным с вычислением различных характеристик СКОРВ. Среди них важное место занимают задачи расчета минимального числа ОУ (процессоров и функциональных устройств в вычислительных системах, станков для обработки деталей, роботов, операторов конвейерных линий и др.), обеспечивающих выполнение заданных объемов вычислений (или работ) за минимальное либо заданное (директивное) время. Заметим, что именно эти задачи с практической точки зрения представляют значительный интерес. Методы решения такого рода задач достаточно проработаны в классической теории расписаний [2]. Однако открытым для исследований остается ряд прикладных задач, связанных с оптимальной организацией распределенных вычислений в СКОРВ.

Следует отметить, что для класса однородных сосредоточенных конкурирующих процессов при макроконвейерной либо конвейерной реализации вычислений задачи нахождения минимального числа ОУ решены в [3, 4]. До настоящего времени остаются нерешенными задачи расчета оптимального числа ОУ, обеспечивающих директивное либо минимальное время выполнения заданных объемов вычислений в условиях распределенной обработки неоднородных конкурирующих процессов [5]. В статье предлагаются алгоритмы для решения таких задач.

1. Основные понятия и математическая модель задачи

Как и в [3–6], под *процессом* будем понимать последовательность блоков (команд, процедур) $Q_1, Q_2, \ldots, Q_s$, для выполнения которых используется множество ОУ. При этом процесс называется *распределенным*, если все блоки или часть из них обрабатываются разными ОУ. Для ускорения выполнения процессы могут обрабатываться параллельно.

Понятие *ресурса* применяется для обозначения любых объектов вычислительной системы, которые могут быть использованы процессами для своего выполнения. *Реентерабельные* (многократно применяемые) ресурсы характеризуются возможностью одновременного использования несколькими вычислительными процессами. Для СКОРВ характерной является ситуация, когда одну и ту же последовательность блоков или ее часть необходимо выполнять многократно несколькими ОУ; такую последовательность будем называть *программным ресурсом*, а соответствующие процессы – *конкурирующими.*

Математическая модель СКОРВ включает: $p\,(p \geq 2)$ – число ОУ; $s\,(s \geq 2)$ – число блоков линейно структурированного программного ресурса $PR = (Q_1, Q_2, \ldots, Q_s)$; $n\,(n \geq 2)$ –

число распределенных относительно *PR* конкурирующих процессов; $T = [t_{ij}]$ – матрицу времен выполнения *j*-х блоков *i*-ми конкурирующими процессами, $i = \overline{1,n}$; $j = \overline{1,s}$.

Так же, как в [3–6], будем считать, что взаимодействие процессов, ОУ и блоков линейно структурированного программного ресурса подчинено следующим условиям:

1) ни один из блоков *PR* не может обрабатываться одновременно более чем одним ОУ;

2) ни один из ОУ не может обрабатывать одновременно более одного блока;

3) обработка каждого блока осуществляется без прерываний;

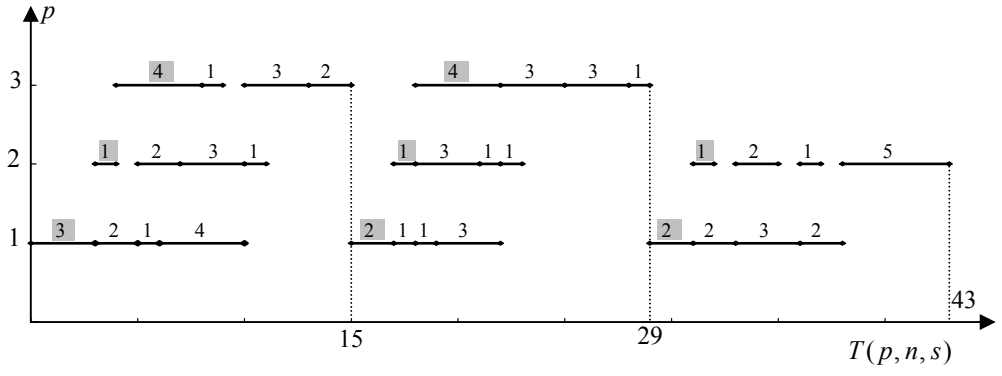
4) распределение блоков программного ресурса по ОУ для каждого из процессов осуществляется циклически по правилу: блок с номером $j = kp + i$ ($j = \overline{1,s}$; $i = \overline{1,p}$; $k \geq 0$) распределяется на ОУ с номером i;

5) отсутствуют простои ОУ при условии готовности блоков, а также невыполнение блоков при наличии ОУ.

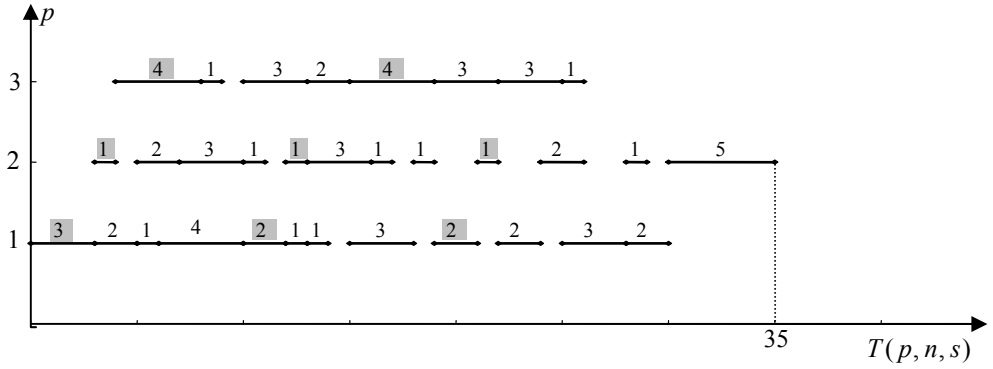
Условия 1–5 определяют *асинхронный режим* взаимодействия ОУ, процессов и блоков, который предполагает отсутствие простоев ОУ при условии готовности блоков, а также невыполнение блоков при наличии свободных ОУ.

СКОРВ называется *неоднородной*, если времена выполнения блоков *PR* зависят от объемов обрабатываемых данных и/или их структуры, т. е. являются разными для разных процессов.

На рис. 1 приведен пример распределенной СКОРВ для асинхронного режима взаимодействия процессов, ОУ и блоков, а также ее отображение с помощью линейных диаграмм Ганта.



а)



б)

Рис. 1. Асинхронный режим: *а)* несовмещенная диаграмма Ганта; *б)* совмещенная

Пусть имеется система из n ($n \geq 2$) неоднородных конкурирующих процессов, готовых к выполнению в начальный момент времени и пронумерованных как 1, 2, ..., *n*. При этом предпо-

лагается, что все n процессов используют только одну копию программного ресурса. Пусть далее $T = [t_{ij}]$ – вещественная ($n \times s$)-матрица, элемент t_{ij} которой определяет время выполнения j-го блока для i-го процесса, где $i = \overline{1,n}$; $j = \overline{1,s}$; d – заданное (директивное) время выполнения всех конкурирующих процессов.

Общее время выполнения всех конкурирующих процессов $T(p, n, s)$ при фиксированных n, s и заданном структурировании программного ресурса, определяемом ($n \times s$)-матрицей $T = [t_{ij}]$, будет существенно зависеть от количества имеющихся ОУ [3, 4].

Задача состоит в том, чтобы при заданных n, s, d и матрице $T = [t_{ij}]$ (где $i = \overline{1,n}$; $j = \overline{1,s}$) найти оптимальное число ОУ, обеспечивающих директивное либо минимальное время выполнения неоднородных конкурирующих процессов.

Для решения поставленной задачи кроме введенных выше параметров математической модели p, n, s, d необходимы $\Gamma^{(q)}$ – двухмерный массив переменной длины, составленный специальным образом из элементов матрицы $T = [t_{ij}]$ (см., например, [6]), q ($q \in N$) – порядковый номер результирующей матрицы времен выполнения блоков (двухмерного массива переменной длины $\Gamma^{(q)}$), а также приведенные ниже определение и теорема.

О п р е д е л е н и е. *Число ОУ в СКОРВ будем называть достаточным и обозначать* p_* *при заданных* n, s, *если* $p_* = s$, *ограниченным, если* $p < s$, *и избыточным, если* $p > s$.

Ниже под критическим путем сетевого вершинно-взвешенного графа понимается путь с наибольшей суммой весов его вершин. Алгоритм его нахождения для линейных графов общеизвестен.

Если $t_{ij} = 0$, то это означает, что все соответствующие вершины графа равны нулю и не влияют на значение критического пути (приводятся для заполнения матрицы).

Пусть $T_*(p_*, n, s)$ – минимальное общее время выполнения множества конкурирующих процессов при достаточном числе ОУ, равном p_*; $T(p, n, s)$ – минимальное общее время при ограниченном или избыточном числе ОУ, равном p.

Теорема. *При заданных* $n, s, T = [t_{ij}]$, $i = \overline{1,n}$, $j = \overline{1,s}$, *в случаях достаточного* ($p_* = s$) *и ограниченного* ($p < s$) *числа ОУ в СКОРВ имеет место соотношение* $T_*(p_*, n, s) \le T(p, n, s)$.

Доказательство следует из того, что при достаточном числе p ОУ минимальное общее время $T_*(p_*, n, s)$ вычисляется как длина критического пути в сетевом вершинно-взвешенном графе, определяемом ($n \times s$)-матрицей $T = [t_{ij}]$. При ограниченном числе $p < s$ ($s = kp + r$, $1 \le r \le p$) ОУ минимальное общее время $T(p < s, n, s)$, как показано в [6], определяется длиной критического пути в сетевом вершинно-взвешенном графе, задаваемом $(k+1)n \times (k+1)p$ -матрицей специального вида

$$\begin{bmatrix} \Gamma_1 & \Gamma_2 & \Gamma_3 & \cdots & \Gamma_k & \Gamma_{k+1} \\ \Gamma_2 & \Gamma_3 & \Gamma_4 & \cdots & \Gamma_{k+1} & Z_0 \\ \Gamma_3 & \Gamma_4 & \Gamma_5 & \cdots & Z_0 & Z_0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \Gamma_k & \Gamma_{k+1} & Z_0 & \cdots & Z_0 & Z_0 \\ \Gamma_{k+1} & Z_0 & Z_0 & \cdots & Z_0 & Z_0 \end{bmatrix}. \qquad (1)$$

В матрице (1) ($n \times p$)-матрицы Γ_l ($l = 1, ..., к+1$) имеют вид

$$\begin{bmatrix} t_{1,(l-1)p+1} & t_{1,(l-1)p+2} & \cdots & t_{1,lp} \\ t_{2,(l-1)p+1} & t_{2,(l-1)p+2} & \cdots & t_{2,lp} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n,(l-1)p+1} & t_{n,(l-1)p+2} & \cdots & t_{n,lp} \end{bmatrix}, \qquad (2)$$

а $(n\times p)$-матрицы Z_0 являются нулевыми. При этом в матрице Γ_{k+1} вида (2) при $r < p$ последние $p - r$ столбцов состоят из нулей.

Ввиду того что исходная $(n\times s)$-матрица $T = [t_{ij}]$ входит в качестве подматрицы в первую строку матрицы (1), сетевой вершинно-взвешенный граф, определяемый $(n\times s)$-матрицей $T = [t_{ij}]$, будет подграфом этого графа, определяемого матрицей (1). Так как длина критического пути в сетевом подграфе данного графа не может превышать такую длину в самом сетевом графе, то справедливо неравенство $T_*(p_*, n, s) \le T(p, n, s)$, что и требовалось доказать.

Следует заметить, что в случае $p < s$ согласно принятой модели [6] достаточно использовать только s ОУ, а остальные $p - s$ не будут задействованы.

Данная теорема является отправной точкой для построения метода нахождения оптимального числа ОУ, обеспечивающих директивное и минимальное время выполнения неоднородных конкурирующих процессов при распределенной обработке в условиях асинхронного режима их взаимодействия в случаях достаточного ($p_* = s$) и ограниченного ($p < s$) чисел ОУ СКОРВ.

Пример 1. На рис. 1 представлены несовмещенная и совмещенная линейные диаграммы Ганта, с помощью которых отражено выполнение $n = 4$ неоднородных распределенных конкурирующих процессов в СКОРВ с $p = 3$ ОУ при $s = 8$ блоках структурированного программного ресурса. Длительности выполнения каждого из блоков указаны на диаграммах, причем для первого процесса они составляют (3, 1, 4, 2, 1, 4, 2, 1), второго – (2, 2, 1, 1, 3, 3, 2, 2), третьего – (1, 3, 3, 1, 1, 3, 3, 1), четвертого – (4, 1, 2, 3, 1, 1, 2, 5). Для наглядности длительности блоков первого процесса выделены. Общее время выполнения $T(p, n, s)$ неоднородных распределенных конкурирующих процессов на несовмещенной диаграмме равно 43, а на совмещенной – 35.

На рис. 2 для данного примера представлен сетевой граф, соответствующий совмещенной линейной диаграмме Ганта (см. рис. 1, *б*). Вершины данного графа содержат соответственно номера процессов (первая цифра), блоков (вторая цифра) и ОУ (третья цифра).

2. Оптимизация числа обрабатывающих устройств при директивном времени

Предлагается следующий алгоритм минимизации числа ОУ при директивном времени.

Входные данные: p – заданное (исходное) число ОУ ($p \ge 2$); n – число конкурирующих неоднородных распределенных процессов ($n \ge 2$); s – число блоков линейно-структурированного программного ресурса ($s \ge 2$); $\Gamma^{(0)}$ – двухмерный массив, содержащий элементы исходной $(n\times s)$-матрицы $T = [t_{ij}]$; d – заданное (директивное) время выполнения конкурирующих процессов.

Выходные данные: $\overline{p}_0$ – минимальное число ОУ, обеспечивающих выполнение конкурирующих процессов за директивное время; $\Gamma^{(q)}$ – двухмерный массив, содержащий результирующую матрицу времен выполнения блоков программного ресурса; q – порядковый номер двухмерного массива, содержащего результирующую матрицу времен выполнения блоков программного ресурса.

Случай 1. Если $d < T_*(p_*, n, s)$, то полагаем $\overline{p}_0 = 0$, т. е. директивное время выполнения конкурирующих процессов d не может быть реализовано в заданных условиях ни для какого числа ОУ.

Случай 2. Пусть $d \geq T_*(p_*, n, s)$ и число ОУ в СКОРВ является ограниченным, т. е. $p < s$. Тогда между d, $T_*(p_*, n, s)$ и $T(p, n, s)$ возможны следующие соотношения:

– если $T_*(p_*, n, s) \leq d = T(p, n, s)$ или $d > T(p, n, \text{s})$, то для нахождения $\overline{p}_0$ используется метод дихотомии, применяемый к отрезку $[2, p]$;

– если $T_*(p_*, n, s) \leq d < T(p, n, s)$, то нахождение $\overline{p}_0$ осуществляется методом дихотомии, применяемым к отрезку $[p, p_*]$.

Случай 3. Пусть $p \geq s$. Тогда нахождение $\overline{p}_0$ осуществляется методом дихотомии, применяемым к отрезку $[2, p_*]$.

Нетрудно подсчитать, что сложность алгоритма нахождения оптимального числа ОУ $\overline{p}_0$, базирующегося на предложенном методе, составляет в худшем случае величину $O((k+1)^2 np \log_2 p)$ операций. Действительно, поиск с использованием алгоритма деления заданного отрезка пополам составляет не более $\log_2 p$ операций, подсчет критического пути в сетевом графе с линейным числом дуг и числом вершин $(k + 1)n \times (\kappa + 1)p$ составляет не более $(k + 1)n \times (\kappa + 1)p$ операций. Таким образом, получаем указанную выше оценку сложности алгоритма нахождения оптимального числа ОУ при заданном директивном времени.

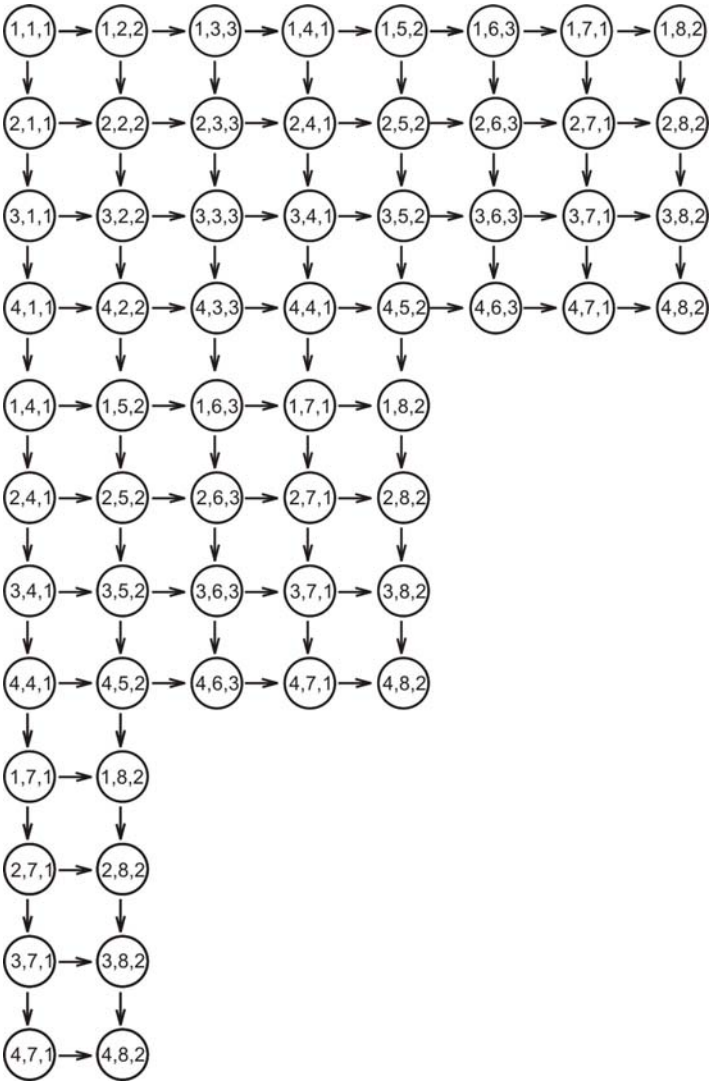


Рис. 2. Сетевой граф для совмещенной линейной диаграммы Ганта

На рис. 3 приведена графическая интерпретация зависимости величины $T(p, n, s)$ от числа p ОУ, а также указаны величины d, $T_*(p_*, n, s)$, $\overline{p}_0$ и p_*. Видно, что величина $\overline{p}_0$ определяется либо как точка пересечения прямой d с дискретной линией, определяющей зависимость $T(p, n, s)$ от p, либо как ближайшая точка, которая находится ниже прямой d.

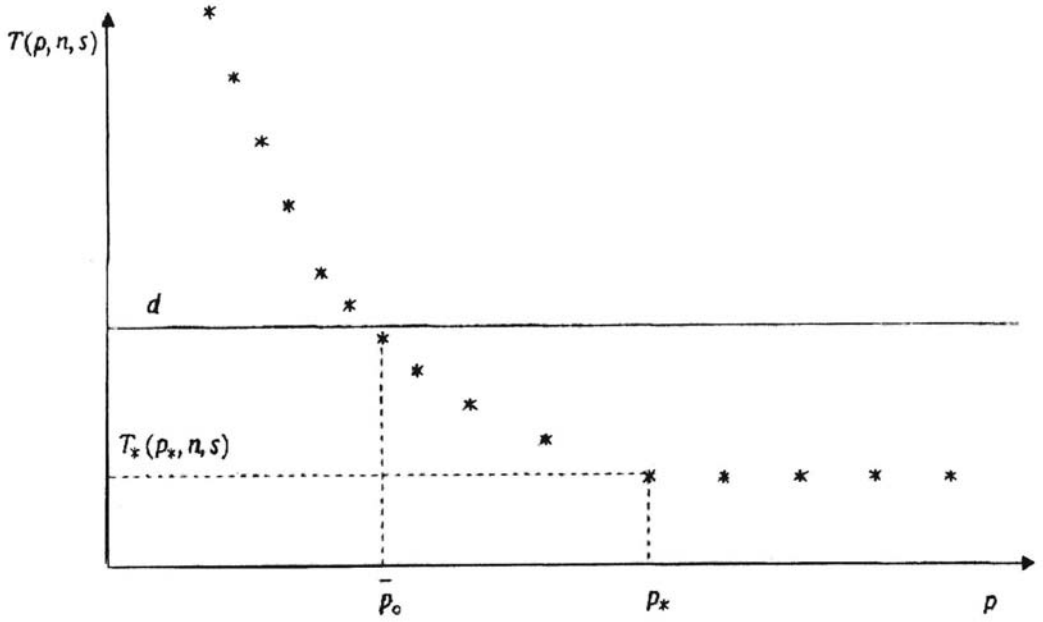


Рис. 3. Графическая интерпретация зависимости величины $T(p, n, s)$ от числа p ОУ

Пример 2. Пусть $d = 50$ – заданное (директивное) время выполнения конкурирующих процессов, $p = 3$ – число ОУ в СКОРВ, $n = 3$ – число конкурирующих неоднородных процессов, $s = 9$ – число блоков линейно-структурированного программного ресурса, а исходная матрица времен выполнения блоков программного ресурса имеет вид

$$\Gamma^{(0)} = \begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 & 4 & 8 & 6 \\ 1 & 7 & 3 & 2 & 5 & 8 & 3 & 1 & 2 \\ 6 & 5 & 2 & 4 & 9 & 3 & 6 & 1 & 3 \end{bmatrix}.$$

В данном случае достаточное число ОУ $p_* = 9$.

1. По исходной матрице $\Gamma^{(0)}$ строим сетевой вершинно-взвешенный граф и находим величину $T_*(p_* = 9, n, s) = 48$ (длину критического пути) [6]. Далее по исходным значениям p, n, s и $\Gamma^{(0)}$ строим результирующую матрицу вида (1), дополненную матрицей вида (2):

$$\begin{bmatrix} \begin{bmatrix} 5 & 2 & 4 \\ 1 & 7 & 3 \\ 6 & 5 & 2 \end{bmatrix} & \begin{bmatrix} 6 & 1 & 7 \\ 2 & 5 & 8 \\ 4 & 9 & 3 \end{bmatrix} & \begin{bmatrix} 4 & 8 & 6 \\ 3 & 1 & 2 \\ 6 & 1 & 3 \end{bmatrix} \\ \begin{bmatrix} 6 & 1 & 7 \\ 2 & 5 & 8 \\ 4 & 9 & 3 \end{bmatrix} & \begin{bmatrix} 4 & 8 & 6 \\ 3 & 1 & 2 \\ 6 & 1 & 3 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 & 6 \\ 3 & 1 & 2 \\ 6 & 1 & 3 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

С помощью полученной матрицы вычисляем величину $T(p = 3,\ n,\ s) = 53$. Учитывая, что $T_*(p_* = 9,\ n,\ s) = 48 \le d = 50 < T(p = 3,\ n,\ s) = 53$, рассмотрим отрезок [3, 9].

2. Методом деления отрезка [3, 9] пополам находим $p_1 = 6$ и строим по заданным n, s и полученному значению $p_1 = 6$ результирующую матрицу $\Gamma^{(1)}$ вида (1), (2):

$$\Gamma^{(1)} = \begin{bmatrix} \begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 \\ 1 & 7 & 3 & 2 & 5 & 8 \\ 6 & 5 & 2 & 4 & 9 & 3 \end{bmatrix} & \begin{bmatrix} 4 & 8 & 6 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 3 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 & 6 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 3 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

С помощью полученной матрицы $\Gamma^{(1)}$ вычисляем величину $T(p_1 = 6,\ n,\ s) = 48$. Так как $T(p_1 = 6,\ n,\ s) = 48 \le d = 50$, то рассматриваем отрезок [3, 6].

3. Методом деления отрезка [3, 6] пополам находим $p_2 = 4$, причем в качестве p_2 берем величину, которая является наименьшим целым полусуммы чисел 3 и 6. Далее по заданным n, s и полученному значению $p_2 = 4$ строим результирующую матрицу $\Gamma^{(2)}$:

$$\Gamma^{(2)} = \begin{bmatrix} \begin{bmatrix} 5 & 2 & 4 & 6 \\ 1 & 7 & 3 & 2 \\ 6 & 5 & 2 & 4 \end{bmatrix} & \begin{bmatrix} 1 & 7 & 4 & 8 \\ 5 & 8 & 3 & 1 \\ 9 & 3 & 6 & 1 \end{bmatrix} & \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 & 4 & 8 \\ 5 & 8 & 3 & 1 \\ 9 & 3 & 6 & 1 \end{bmatrix} & \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

С помощью полученной матрицы $\Gamma^{(2)}$ вычисляем величину $T(p_2 = 4,\ n,\ s) = 49$. Таким образом, директивное время выполнения $n = 3$ процессов реализуется при $p_2 = 4$, так как $d = 50 > T(p_2 = 4,\ n,\ s) = 49$, и не реализуется при $p = 3$, так как $d = 50 < T(p = 3,\ n,\ s) = 53$. Следовательно, $\overline{p}_0 = 4$.

3. Оптимизация числа обрабатывающих устройств при минимальном времени

Предлагается следующий алгоритм минимизации числа ОУ при минимальном времени.

Входные данные: p – заданное (исходное) число ОУ ($p \ge 2$); n – число конкурирующих неоднородных распределенных процессов ($n \ge 2$); s – число блоков линейно-структурированного программного ресурса ($2 \le s \le p$); $\Gamma^{(0)}$ – двухмерный массив, содержащий элементы исходной ($n \times s$)-матрицы $T = [t_{ij}]$; q – порядковый номер двухмерного массива, содержащего результирующую матрицу времен выполнения блоков программного ресурса.

Выходные данные: $T_*(p_*,\ n,\ s)$ – минимально возможное время выполнения програм-много ресурса конкурирующими процессами; $\overline{p}_0$ – минимальное число ОУ, обеспечивающих минимально возможное время выполнения программного ресурса конкурирующими процессами.

Случай 1 ($s \le p$)

1. Находим достаточное число ОУ ($p = s$) и вычисляем минимальное общее время (т. е. $T_*(p_*, n, s)$) при достаточном числе ОУ.

2. Уменьшаем число ОУ p на единицу (начиная с $p = s$) до тех пор, пока оно не станет меньше двух или пока время $T(p, n, s)$, полученное при новом значении p, не станет больше $T_*(p_*, n, s)$.

Случай 2 ($s > p$)

1. Находим достаточное число ОУ ($p = s$) и вычисляем минимальное общее время при достаточном числе ОУ, т. е. $T_*(p_*, n, s)$.

2. Вычисляем $T(p, n, s)$ – время выполнения блоков программного ресурса при заданном числе ОУ p:

– если $T(p, n, s) = T_*(p_*, n, s)$, то уменьшаем исходное (а не достаточное) число ОУ на единицу до тех пор, пока p не станет меньше двух или пока время $T(p, n, s)$ не станет больше чем $T_*(p_*, n, s)$;

– если $T(p, n, s) > T_*(p_*, n, s)$, то увеличиваем число ОУ на единицу до тех пор, пока время $T(p, n, s)$ не станет равным $T_*(p_*, n, s)$.

Пример 3. Пусть $p = 9$ – число ОУ в СКОРВ, $n = 3$ – число конкурирующих неоднородных процессов, $s = 9$ – число блоков линейно-структурированного программного ресурса. Пусть также задана исходная матрица времен выполнения блоков программного ресурса:

$$\Gamma^{(0)} = \begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 & 4 & 8 & 6 \\ 1 & 7 & 3 & 2 & 5 & 8 & 3 & 1 & 2 \\ 6 & 5 & 2 & 4 & 9 & 3 & 6 & 1 & 3 \end{bmatrix}.$$

С помощью приведенного выше метода находим минимальное число ОУ $\overline{p}_0$, обеспечивающих минимально возможное время выполнения конкурирующих процессов:

1. Определяем достаточное число ОУ, обрабатывающих блоки программного ресурса: $p = 9,\ q = 0.$

2. Вычисляем длину критического пути в ориентированном вершинно-взвешенном графе, определяемом матрицей $\Gamma^{(0)}$: $T_*(p = 9, n, s) = 48$.

3. Уменьшаем число ОУ на единицу: $p = 8,\ p > 2,\ q = 1.$

4. Строим результирующую матрицу $\Gamma^{(1)}$ размерности 6×16 по заданным $n = 3, s = 9$ и новому значению $p = 8$:

$$\Gamma^{(1)} = \left[\begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 & 4 & 8 \\ 1 & 7 & 3 & 2 & 5 & 8 & 3 & 1 \\ 6 & 5 & 2 & 4 & 9 & 3 & 6 & 1 \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right].$$

5. Вычисляем длину критического пути в орграфе, определяемом матрицей $\Gamma^{(1)}$: $T(p = 8, n, s) = 48$, $T(p = 8, n, s) = T_*(p = 9, n, s) = 48$.

6. Уменьшаем число ОУ на единицу: $p = 7,\ p > 2,\ q = 2.$

7. Строим результирующую матрицу $\Gamma^{(2)}$ размерности 6×14 по заданным $n = 3, s = 9$ и новому значению $p = 7$:

$$\Gamma^{(2)} = \left[\begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 & 4 \\ 1 & 7 & 3 & 2 & 5 & 8 & 3 \\ 6 & 5 & 2 & 4 & 9 & 3 & 6 \end{bmatrix} \begin{bmatrix} 8 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} 8 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right].$$

8. Вычисляем длину критического пути в орграфе, определяемом матрицей $\Gamma^{(2)}$: $T(p=7,\ n,\ s)=48$, $T(p=7,\ n,\ s)=T_*(p=9,\ n,\ s)=48$.

9. Уменьшаем число ОУ на единицу: $p=6,\ p>2,\ q=3$.

10. Строим результирующую матрицу $\Gamma^{(3)}$ размерности 6×12 по заданным $n=3,\ s=9$ и новому значению $p=6$:

$$\Gamma^{(3)} = \left[\begin{bmatrix} 5 & 2 & 4 & 6 & 1 & 7 \\ 1 & 7 & 3 & 2 & 5 & 8 \\ 6 & 5 & 2 & 4 & 9 & 3 \end{bmatrix} \begin{bmatrix} 4 & 8 & 6 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 3 & 0 & 0 & 0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} 4 & 8 & 6 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 3 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right].$$

11. Вычисляем длину критического пути в орграфе, определяемом матрицей $\Gamma^{(3)}$: $T(p=6,\ n,\ s)=48$, $T(p=6,\ n,\ s)=T_*(p=9,\ n,\ s)=48$.

12. Уменьшаем число ОУ на единицу: $p=5,\ p>2,\ q=4$.

13. Строим результирующую матрицу $\Gamma^{(4)}$ размерности 6×10 по заданным $n=3,\ s=9$ и новому значению $p=5$:

$$\Gamma^{(4)} = \left[\begin{bmatrix} 5 & 2 & 4 & 6 & 1 \\ 1 & 7 & 3 & 2 & 5 \\ 6 & 5 & 2 & 4 & 9 \end{bmatrix} \begin{bmatrix} 7 & 4 & 8 & 6 & 0 \\ 8 & 3 & 1 & 2 & 0 \\ 3 & 6 & 1 & 3 & 0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} 7 & 4 & 8 & 6 & 0 \\ 8 & 3 & 1 & 2 & 0 \\ 3 & 6 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right].$$

14. Вычисляем длину критического пути в орграфе, определяемом матрицей $\Gamma^{(4)}$: $T(p=5,\ n,\ s)=48$, $T(p=5,\ n,\ s)=T_*(p=9,\ n,\ s)=48$.

15. Уменьшаем число ОУ на единицу: $p=4,\ p>2,\ q=5$.

16. Строим результирующую матрицу $\Gamma^{(5)}$ размерности 9×12 по заданным $n=3,\ s=9$ и новому значению $p=4$:

$$\Gamma^{(5)} = \left[\begin{bmatrix} 5 & 2 & 4 & 6 \\ 1 & 7 & 3 & 2 \\ 6 & 5 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 7 & 4 & 8 \\ 5 & 8 & 3 & 1 \\ 9 & 3 & 6 & 1 \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} \right.$$
$$\begin{bmatrix} 1 & 7 & 4 & 8 \\ 5 & 8 & 3 & 1 \\ 9 & 3 & 6 & 1 \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\left. \begin{bmatrix} 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right].$$

17. Вычисляем длину критического пути в орграфе, определяемом матрицей $\Gamma^{(5)}$: $T(p=4,\ n,\ s)=49$, $T(p=4,\ n,\ s)>T_*(p=9,\ n,\ s)$.

18. $\overline{p}_0=5$.

Таким образом, минимально возможное время выполнения программного ресурса конкурирующими процессами $T_*(p_*,\ n,\ s)=48$, а минимальное число ОУ, обеспечивающих выполнение программного ресурса за это время, равно 5.

Пример 4. Пусть $p=3$ – число ОУ в СКОРВ, $n=4$ – число конкурирующих неоднородных процессов, $s=9$ – число блоков линейно-структурированного программного ресурса. Пусть также задана исходная матрица времен выполнения блоков программного ресурса:

$$\Gamma^{(0)}=\begin{bmatrix}2&5&1&4&7&2&3&8&1\\6&3&2&8&5&1&7&2&3\\3&7&5&2&6&3&1&4&6\\1&6&2&3&1&5&6&1&2\end{bmatrix}.$$

Результирующая матрица $\Gamma^{(1)}$ времен выполнения блоков программного ресурса в этом случае будет иметь вид

$$\Gamma^{(1)}=\left[\begin{bmatrix}2&5&1\\6&3&2\\3&7&5\\1&6&2\end{bmatrix}\begin{bmatrix}4&7&2\\8&5&1\\2&6&3\\3&1&5\end{bmatrix}\begin{bmatrix}3&8&1\\7&2&3\\1&4&6\\6&1&2\end{bmatrix}\right.$$
$$\left.\begin{bmatrix}4&7&2\\8&5&1\\2&6&3\\3&1&5\end{bmatrix}\begin{bmatrix}3&8&1\\7&2&3\\1&4&6\\6&1&2\end{bmatrix}\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\\0&0&0\end{bmatrix}\right.$$
$$\left.\begin{bmatrix}3&8&1\\7&2&3\\1&4&6\\6&1&2\end{bmatrix}\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\\0&0&0\end{bmatrix}\right].$$

Находим минимальное число $\overline{p}_0$ ОУ, обеспечивающих выполнение множества конкурирующих процессов за минимальное время. Из теоремы следует, что это время не может превышать минимального общего времени выполнения конкурирующих процессов при достаточном числе ОУ:

1. $q=1$.

2. Находим длину критического пути в орграфе, определяемом исходной матрицей $\Gamma^{(0)}$: $T_*(p=9,\ n,\ s)=46$.

3. Вычисляем длину критического пути в орграфе, определяемом результирующей матрицей $\Gamma^{(1)}$: $T(p=3,n,s)=3$, $T(p=3,n,s)>T_*(p=9,\ n,\ s)$.

4. Увеличиваем на единицу число ОУ, обрабатывающих программный ресурс: $p=4$, $q=2$.

5. Строим новую результирующую матрицу $\Gamma^{(2)}$ размерности 12×12 по заданным n, s и новому значению $p=4$:

$$\Gamma^{(2)} = \begin{bmatrix} \begin{bmatrix} 2 & 5 & 1 & 4 \\ 6 & 3 & 2 & 8 \\ 3 & 7 & 5 & 2 \\ 1 & 6 & 2 & 3 \end{bmatrix} & \begin{bmatrix} 7 & 2 & 3 & 8 \\ 5 & 1 & 7 & 2 \\ 6 & 3 & 1 & 4 \\ 1 & 5 & 6 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 & 3 & 8 \\ 5 & 1 & 7 & 2 \\ 6 & 3 & 1 & 4 \\ 1 & 5 & 6 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

6. Находим длину критического пути в орграфе, определяемом матрицей $\Gamma^{(2)}$: $T(p=4, n, s) = 52$, $T(p=4, n, s) > T_*(p=9, n, s)$.

7. Увеличиваем число ОУ, обрабатывающих программный ресурс, на единицу: $p=5,\ q=3$.

8. Строим новую результирующую матрицу $\Gamma^{(3)}$ размерности 8×10 по заданным n, s и новому значению $p=5$:

$$\Gamma^{(3)} = \begin{bmatrix} \begin{bmatrix} 2 & 5 & 1 & 4 & 7 \\ 6 & 3 & 2 & 8 & 5 \\ 3 & 7 & 5 & 2 & 6 \\ 1 & 6 & 2 & 3 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 3 & 3 & 1 & 0 \\ 1 & 7 & 7 & 3 & 0 \\ 3 & 1 & 1 & 6 & 0 \\ 5 & 6 & 6 & 2 & 0 \end{bmatrix} \\ \begin{bmatrix} 2 & 3 & 3 & 1 & 0 \\ 1 & 7 & 7 & 3 & 0 \\ 3 & 1 & 1 & 6 & 0 \\ 5 & 6 & 6 & 2 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

9. Находим длину критического пути в орграфе, определяемом матрицей $\Gamma^{(3)}$: $T(p=5, n, s) = 48$, $T(p=5, n, s) > T_*(p=9, n, s)$.

10. $p_0 = 5$. Выход.

Итак, минимальное число ОУ, обеспечивающих выполнение программного ресурса за минимальное время $T_*(p=9, n, s) = 49$, равно 5. Выигрыш по времени в результате применения алгоритма составляет 16 единиц (или 24,6 %).

Заключение

В статье исследован ряд прикладных задач, связанных с оптимальной организацией распределенных вычислений в системах с конвейерной организацией распределенных вычислений. Для данных систем построена математическая модель организации неоднородных распределенных конкурирующих процессов при условии асинхронного режима их взаимодействия. Для нахождения минимального числа обрабатывающих устройств при выполнении заданных объемов вычислений за директивное либо минимальное время предложены алгоритмы, которые имеют полиномиальную сложность, что особенно важно для их практических приложений. Работа алгоритмов наглядно подтверждена конкретными примерами, которые показывают их эффективность и возможность простого и удобного использования на практике.

Список литературы

1. Абламейко, С.В. Принципы построения суперкомпьютеров семейства СКИФ и их реализация / С.В. Абламейко [и др.] // Информатика. – 2004. – № 1. – С. 89–106.

2. Танаев, В.С. Теория расписаний. Групповые технологии / В.С. Танаев, М.Я. Ковалев, Я.М. Шафранский. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1998. – 290 с.

3. Капитонова, Ю.В. Метод структурирования программных ресурсов при организации параллельных вычислений / Ю.В. Капитонова, Н.С. Коваленко // Докл. IV Всесоюз. симпоз. «Системное и теоретическое программирование». – Кишинев : Штиинца, 1983. – С. 183–185.

4. Овсеец, М.И. Минимизация числа обрабатывающих устройств при реализации однородных конкурирующих процессов / М.И. Овсеец // Доклады АН БССР. – 1985. – № 12. – С. 1082–1085.

5. Иванников, В.П. О минимальном времени реализации конкурирующих процессов в синхронных режимах / В.П. Иванников, Н.С. Коваленко, В.М. Метельский // Программирование. – 2000. – № 5. – С. 268–274.

6. Коваленко, Н.С. О времени реализации конкурирующих процессов при распределенной обработке / Н.С. Коваленко, В.М. Метельский // Кибернетика и системный анализ. – 1996. – № 1. – С. 54–64.

Поступила 07.02.2013

[1]*Белорусский государственный экономический университет,*
Минск, пр. Партизанский, 26
e-mail: kovalenkons@rambler.ru

[2]*Объединенный институт проблем информатики НАН Беларуси,*
Минск, ул. Сурганова, 6
e-mail: vengerov@basnet.by

[3]*Белорусский государственный университет информатики и радиоэлектроники,*
Минск, ул. П. Бровки, 6
e-mail: metmargen@mail.ru

N.S. Kovalenko, V.N. Vengherov, V.M. Metelyskij

MINIMIZING THE NUMBER OF PROCESSING UNITS IN DISTRIBUTED COMPUTATIONS

Polynomial algorithms for finding the minimum number of processing units in the problem of scheduling heterogeneous competing processes on parallel processors are developed. The algorithms ensure minimum completion time of all the processes and their completion by the specified deadline.