2.3 Постановка задачи, математическая постановка алгоритмов отказоустойчивого распределения задач

За основу взят критерий интенсивности размещения, значение которого показывает насколько оптимально произведено распределение задач по процессорным модулям системы. Критерий описывает взаимодействие подпрограмм в мультипроцессорной системе [1,2]. Подпрограммы назначаются на смежные процессорные элементы, повышая производительность системы и уменьшая задержки передачи информации между процессорами. Задача состоит в построении лучшего маршрута передачи данных в параллельной системе. В случае отказа производится оптимальное перераспределение подпрограмм в мультипроцессорной системе.

Исходная программа представлена графом. Вершины графа представляют участки, дуги показывают зависимость по данным. После преобразования исходной программы, данные представлены в виде множества задач, которые нужно оптимально распределить по процессорным модулям кубической мультипроцессорной системы.

При отказе одного и более процессоров инициируется процедура перераспределения задач по процессорным модулям и дальнейшей оптимизации маршрутизации сообщений по смежным процессорам для увеличения производительности и уменьшения нагрузки на систему.

Обобщенный алгоритм планирования задач в параллельной системе с учетом отказов состоит из следующих шагов [1-3]:

Получить граф задач.

Распределить задачи в кубической топологии.

Перераспределить задачи по процессорным модулям в случае отказа процессора.

Оптимизировать планирование задач в мультипроцессорной системе для уменьшения времени обработки данных.

Для повышения надежности системы и обеспечения более высокой производительности будет использоваться замена отказавшего процессора резервным.

Математическая постановка алгоритма отказоустойчивого распределения задач. Каждая из реализуемых системой программ представляется множеством взаимосвязанных подпрограмм. Множество подпрограмм каждой программы описывается графом взаимодействия задач:

$$F=(X,E) \quad (1)$$

где $X=\begin{pmatrix} x_{1.1} & x_{1.2} & \dots & x_{1.k} & \dots & x_{1.n} \\ x_{2.1} & x_{2.2} & \dots & x_{2.k} & \dots & x_{2.n} \\ \dots & & & & & \\ x_{q.1} & x_{q.1} & \dots & x_{q.k} & \dots & x_{q.n} \\ \dots & & & & & \\ x_{n.1} & x_{n.2} & \dots & x_{n.k} & \dots & x_{n.n} \end{pmatrix}$ – множество вершин, которые соответствуют отдельным программам, $E=\{e_{ij}\}$ – множество дуг, отражающих связи между программами.

Граф F представляется матрицей обмена информации: $IEM=||m_{ij}||_{N\times N}$, где N – количество вершин графа F, а m_{ij} – число соответствующее значению объема данных, передаваемых между подпрограммами x_i и x_j.

Мультипроцессорная система представляется графом A=(P,V), где $P=\begin{pmatrix} p_{1.1} & p_{1.2} & \dots & p_{1.n} \\ p_{2.1} & p_{2.2} & \dots & p_{2.n} \\ \dots & & & \\ p_{n.1} & p_{n.2} & \dots & p_{n.n} \end{pmatrix}$ – множество идентификаторов процессорных модулей, организованных в матрицу $|P|_{n\times n}$, где мощность $|P| = N = n^2$ – число процессорных модулей системы; V – множество межмодульных связей, задаваемых матрицей смежности $=||M||_{N\times N}$ размером $n^2\times n^2$.

Размещение пакета программ (комплекса задач), описываемых графом F может быть аналитически описано отображением

$$\alpha_S=\begin{pmatrix} x_{S_{1.1}} & x_{S_{1.2}} & \dots & x_{S_{1.k}} & \dots & x_{S_{1.n}} \\ x_{S_{2.1}} & x_{S_{2.2}} & \dots & x_{S_{2.k}} & \dots & x_{S_{2.n}} \\ \dots & & & & & \\ x_{S_{q.1}} & x_{S_{q.1}} & \dots & x_{S_{q.k}} & \dots & x_{S_{q.n}} \\ \dots & & & & & \\ x_{S_{n.1}} & x_{S_{n.2}} & \dots & x_{S_{n.k}} & \dots & x_{S_{n.n}} \end{pmatrix} \rightarrow \begin{pmatrix} p_{1.1} & p_{1.2} & \dots & p_{1.k} & \dots & p_{1.n} \\ p_{2.1} & p_{2.2} & \dots & p_{2.k} & \dots & p_{2.n} \\ \dots & & & & & \\ p_{q.1} & p_{q.2} & \dots & p_{q.k} & \dots & p_{q.n} \\ \dots & & & & & \\ p_{n.1} & p_{n.2} & \dots & p_{n.k} & \dots & p_{n.n} \end{pmatrix}, \quad (2)$$

где $s = \overline{1, N!},\ k = \overline{1, n}, q = \overline{1, n}.$

Здесь s – это номер очередной перестановки, соответствующий s-му варианту размещения. Мощность множества $\omega = \{\alpha_S\}$ всевозможных отображений (2) равна числу всевозможных перестановок подпрограмм $\{x_{qk}\}$ в матрице X : |ω| = N!. Введем также матрицу расстояний G = $||d_{ij}||_{N\times N}$, $N=n^2=|H|$, символ «⟶» означает отображение подпрограммы $x_{S_{n.n}}$ на процессор $p_{n,n}$.

Кубическая топология мультипроцессорной системы задается графом H=(S_1,D), где множество вершин S_1 соответствует процессорным модулям, а множество дуг D – межмодульным связям. Множество S_1 разбивается на два непересекающихся подмножества S_1 = S∪L, где S={s_{ij}} – множество основных процессоров, L={l_{ij}} – множество резервных процессоров, причем фиксируется |S|= n^3 и |L|=n^2, n=2,3,4,…. Упорядочим множества процессоров S и L в виде матриц $||s_{ij}||_{n\times n}$ и $||l_{ij}||_{n\times n}$ соответственно. Множество S_1 представим объединением указанных матриц следующим образом:

$$\begin{cases} \begin{pmatrix} s_{v.1.1} & s_{v.1.2} & \dots & s_{v.1.n} \\ s_{v.2.1} & s_{v.2.1} & \dots & s_{v.2.1} \\ \dots & & & \\ s_{v.n.1} & s_{v.n.2} & \dots & s_{v.n.n} \end{pmatrix} \\ \begin{pmatrix} l_{v.1.1} & l_{v.1.2} & \dots & l_{v.1.n} \\ l_{v.2.1} & l_{v.2.1} & \dots & l_{v.2.1} \\ \dots & & & \\ l_{v.n.1} & l_{v.n.2} & \dots & l_{v.n.n} \end{pmatrix} \end{cases} \quad (3)$$

Топология кубической мультипроцессорной системы с учетом резервных модулей представлена на рисунке 1.

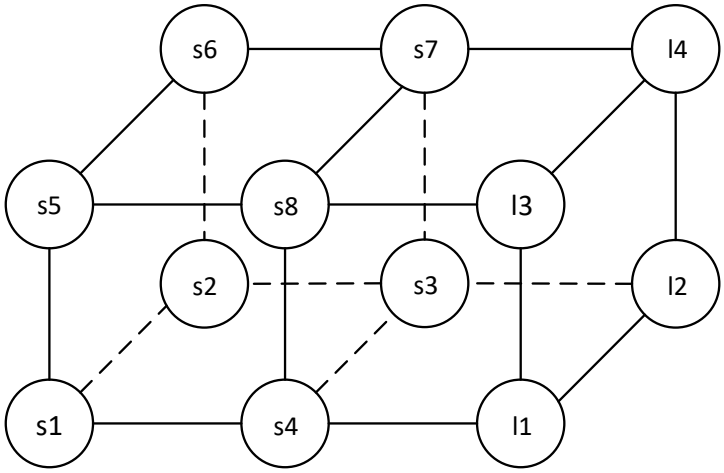


Рисунок 1 - Кубическая организация мультипроцессорной системы с резервными модулями

Размещение множества взаимосвязанных программ, описываемого графом F, в мультипроцессорной системе задается отображением:

$$\alpha: X \rightarrow S_1, \quad (4)$$

которое ставит в соответствие каждой подпрограмме один из процессоров

Между каждой парой процессоров матрицы в общем случае существует множество маршрутов передачи сообщений, которые обеспечивают разное время доставки данных между соответствующими подпрограммами. В рамках решаемой задачи интерес представляют кратчайшие маршруты, гарантирующие минимум указанного времени. Для описания множества таких маршрутов вводится матрица минимальных расстояний $MRM=\|d_{ij}\|_{N\times N}$, элемент d_{ij}, который численно равен длине кратчайшего пути между процессорными модулями, в которых размещены подпрограммы x_i и x_j.

Пусть γ – множество всевозможных отображений. Тогда задача размещения программ в мультипроцессорных системах заключается в выборе такого отображения $\alpha_g \in \gamma$, которое соответствует следующему критерию:

$$\xi_g = \min_{\alpha \in \gamma} \left\{ \max_{i=1,N,j=1,N} \{ d_{ij} \times m_{ij} \} \right\}, \quad (5)$$

где максимум в фигурных скобках представляет собой наибольшую частную коммуникационную задержку для заданного отображения α.

Сущность созданного алгоритма заключается в двухэтапном решении задачи размещения. На первом этапе вычисляется теоретически минимальное наибольшее частное значение интенсивности размещения ξ_{min} из предположения, что граф F размещается в мультипроцессорной системе без учета ограничений связности подпрограмм (дуги графа F «накладываются» наилучшим образом, то есть так, что более высоким значениям m_{ij} соответствуют меньшие длины маршрутов d_{ij}). На втором этапе отыскивается

вариант размещения αg, который минимизирует отклонение $|\xi_g - \xi_{min}|$ и, таким образом, дает локальный минимум наибольшей коммуникационной задержки [1].

В случае наличия в мультипроцессорной системе отказавших процессоров размещение вычисления по формуле (5) для различных отображений вида (4) проводятся с учетом неоднородности топологической структуры.

В этом случае учитывается, что отказавший процессор замещается резервным, например, процессором l_3 и изменяется множество допустимых маршрутов передачи данных (см. рисунок 2).

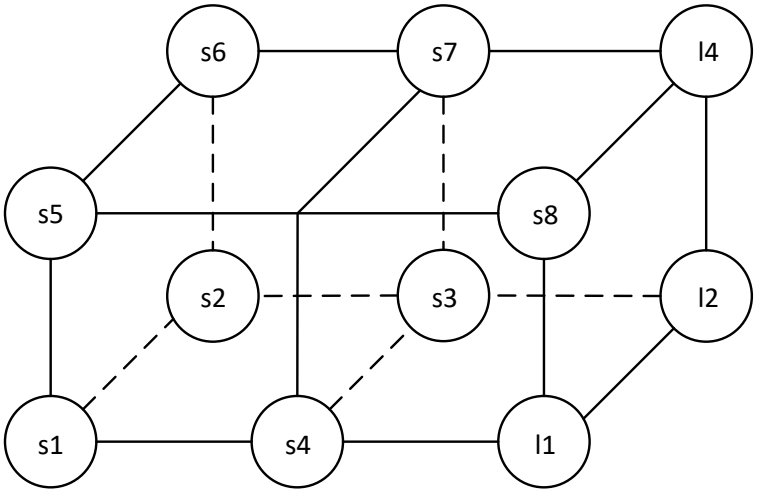


Рисунок 2 – Кубическая организация мультипроцессорной системы с отказавшим модулем s_8

Процедура замещения отказавшего процессорного модуля резервным формализуется следующим образом.

Состояние основных процессоров кубической системы описывается матрицей $Q=||q_{ij}||_{n\times n}$. При $q_{ij}=0$ процессорный модуль считается неисправным и при $q_{ij}=1$ процессор исправен.

Исправность резервных процессоров описывается матрицей $U=||u_{ij}||_{n\times n}$. По аналогии: $u_{ij}=0$ – неисправен, $u_{ij}=1$ – исправен.

Матрица $W=\|w_{ij}\|_{n\times n}$ содержит признаки доступности ближайшего резервного процессорного модуля для замены неисправного основного: $w_{ij}=0$ – недоступен, $w_{ij}=1$ – доступен.

Алгоритм замены отказавшего процессора на резервный. Описание алгоритма замены отказавшего процессорного модуля резервным заключается в следующих этапах:

1. Загружаются матрица обмена информации IEM и матрица минимальных расстояний MRM.

2. Устанавливается что $q_{ij}= u_{ij}=w_{ij}=0$, $i=\overline{1,n}, j=\overline{1,n}$.

3. Инициируется поиск по всем процессорам $s_{ij} \in S$ на предмет отказа.

4. В случае отказа основного процессора $q_{ij}=0$, то находится в матрице U элемент $u_{st}=1$ такой, что $(|i-s|\geq 0)\wedge(|j-t|\geq 0)$.

5. В случае если такой u_{st} найден, то принять $u_{st}=0$, $l_{st}=s_{ij}$ (замещение успешно), иначе необходим останов системы.

В предложенном алгоритме предполагается, что при поиске резервного модуля область поиска ограничена ближайшими соседними процессорными модулями [1].

Алгоритм перераспределения задач в случае отказа процессорного модуля. Так как, после замены, отказавшего процессорного модуля резервным, изменяется топологическая организация мультипроцессорной системы, на следующем этапе следует выполнять процедуру перераспределения программ на новый вариант топологической организации мультипроцессорной системы.

Важным этапом созданного метода перераспределения является определение теоретической нижней оценки наибольшего частного значения интенсивности размещения $\xi_{min\ max}$, относительно которой осуществляется минимизация времени межпроцессорного обмена при выборе варианта размещения программ согласно постановке задачи. Для вычисления указанной оценки используется следующий алгоритм.

1. Переписать все значения $d_{ij} \neq 0$ из матрицы MRM в вектор MRM' в порядке возрастания, то есть соблюдая условие: $\forall q_1 \neq q_2$: $d_{i'j'}^{z_1} \geq d_{i'j'}^{z_2} \Leftrightarrow q_1 > q_2$, где q_1, q_2 – порядковые номера значений $d_{i'j'}$, $d_{i'j'}$ в векторе MRM'.

2. Переписать все значения $m_{ij} \neq 0$ из матрицы IEM в вектор IEM' в порядке убывания, то есть соблюдая условие: $\forall q_1 \neq q_2$: $m_{i'j'}^{z_1} \leq m_{i'j'}^{z_2} \Leftrightarrow q_1 > q_2$, где q_1, q_2 – порядковые номера значений $m_{i'j'}$, $m_{i'j'}$ в векторе IEM'.

3. По векторам MRM' и IEM' вычислить критерий интенсивности размещения $\xi_{min\ max}$, используя формулу:

$$\xi_{minmax} = \max_q \{m_{ij}^q d_{ij}^q\}, \quad (4)$$

где m_{ij}^q, d_{ij}^q – элементы векторов MRM' и IEM', расположенные в одинаковых позициях.

В дальнейшем перераспределение сводится к выполнению нескольких перестановок строк и столбцов матрицы IEM с тем, чтобы достичь наибольшего приближения к оценке $\xi_{min\ max}$. После каждой перестановки вычисляется значение $\max\{m_{ij} \times d_{ij}\}$. Если найденное значение больше предыдущего, то перестановки продолжаются от предыдущего варианта, иначе далее рассматривается текущий вариант [1-3].

С учетом вышеизложенного алгоритм отказоустойчивого перераспределения программ в мультипроцессорной системе будет состоять из следующих шагов:

1. Загрузить IEM и MRM;

2. Создать дополнительные матрицы M_1, M_2, M_3 размера N×N для хранения новых вариантов размещения после перестановок строк и столбцов матрицы IEM: M_1 – матрица, где фиксируются проверенные элементы матрицы M_2, а M_3 – это промежуточная матрица.

3. Сформировать векторы IEM' и MRM' по матрицам IEM и MRM соответственно.

4. Найти значение критерия $\xi_{min\ max}$ по формуле (4).

5. Вычислить значение критерия $\xi_0 = \max_{i=\overline{1,N},j=\overline{1,N}}\{d_{ij} \times m_{ij}\}$ для первоначального варианта размещения α_0, определяемого матрицами IEM и MRM.

6. Рассчитать отношение $\sigma_0 = \frac{\xi_0}{\xi_{minmax}}$, если $\sigma_0 \leq \overline{\sigma}$ (где $\overline{\sigma} \geq 1$ – пороговое значение, определяющее точность решения задачи), то останов.

7. Принять $\xi=\xi_0$ и M_2=IEM.

8. Начать просмотр элементов вектора IEM', начиная с элемента k=1.

9. Пусть k-м элементом вектора IEM' является $m_{\mu\varphi}$. Найти элемент $m_{\mu\varphi}$ в матрице M_2, определяя его положение по индексам μ, φ.

10. Найти соответствующую элементу $m_{\mu\varphi}$ длину маршрута $d_{\mu\varphi}$ в матрице MRM.

11. Если $d_{\mu\varphi}$=1, то (так как улучшить размещение нельзя) зафиксировать текущий элемент в M_2 как просмотренный, приняв $M_{2\mu\varphi}$=-1. Принять k=k+1 и если вектор IEM' не просмотрен до конца, то вернуться к п.9, иначе перейти к п.12.

12. Начать просмотр строки d_μ матрицы MRM, с элемента j=1.

13. Если j≥N, строка просмотрена, перейти к п.24, иначе - п.14.

14. Если $d_{\mu j}<d_{\mu\varphi}$ и $d_{\mu j}\neq 0$, то перейти к п.15, иначе принять j=j+1 и вернуться к п.13.

15. Выполнить парную перестановку строк/столбцов i↔φ в матрице IEM и сформировать новую матрицу M_1.

16. Вычислить по матрице M_1 значение критерия $\xi_1 = \max_{i=\overline{1,N},j=\overline{1,N}}\{d_{ij} \times m_{ij}\}$. Если $\xi_1 \leq \xi_0$, то перейти к п.17, иначе принять j=j+1 и вернуться к п.12.

17. Рассчитать отношение $\sigma = \frac{\xi_0}{\xi_1}$ и если $\sigma \leq \overline{\sigma}$, то останов.

18. Переписать матрицу M_1 в IEM.

19 Принять $\xi_0=\xi_1$.

20 Зафиксировать текущий элемент $m_{\mu\varphi}$ в M_2 как просмотренный, приняв $M_{2\mu\varphi}$=-1.

21. Выполнить парную перестановку строк/столбцов i↔φ в матрице M_2 и сформировать новую матрицу M_3.

22. Переписать M_3 в M_2.

23. Принять k=k+1 и, если вектор IEM’ не просмотрен до конца, вернуться к п.9, иначе считать, что все элементы проверены, далее останов системы.

24 Зафиксировать текущий элемент в M_2 как просмотренный ($M_{2\mu\varphi}$=-1), принять k=k+1. Если вектор IEM’ не просмотрен до конца, то перейти к п.9. Иначе считать, что все элементы вектора просмотрены (останов системы).

В случае отказа отдельного канала связи между процессорными модулями нарушаются маршруты передачи данных и необходимо найти новые кратчайшие маршруты обхода отказавшего канала [1]. Для этого случая обоснована целесообразность использования известного алгоритма Дейкстры.

Требования:

1) На вход устройства подается информация в виде матриц:

– матрица смежности;

– матрицу расстояний;

– матрицу исправности процессоров;

– матрицу работоспособности и занятости резервных процессоров;

– матрицу исправности межпроцессорных связей.

2) Устройство должно функционировать на основе алгоритмов планирования переразмещения и реагировать на различные типы отказов, а именно:

– отказ внутреннего процессорного модуля;

– отказ межпроцессорной связи.

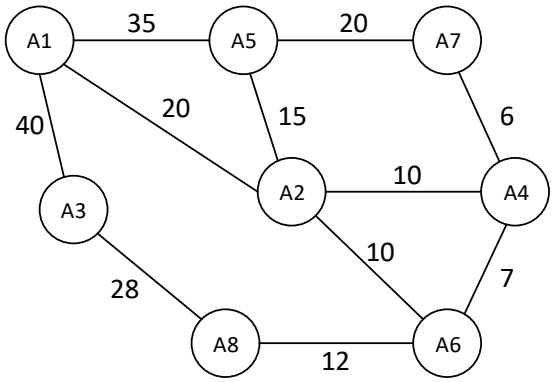
3) Результатом работы является файл конечной матрицы смежности, матрицы расстояний и матрицы исправности межпроцессорных связей, соответствующей найденному варианту размещения и/или кратчайшего маршрута обхода.

4) Устройство должно быть промоделировано, построены графики зависимости временной и аппаратной сложности.

5) Структурная и принципиальная схемы.

Алгоритм размещения задач в кубической топологии мультипроцессорных систем:

1. Задаем случайный взвешенный граф или задаем граф вручную (число вершин не должно превышать число процессоров в заданном кубе)
   Пример (дальше будет идти по этому примеру):



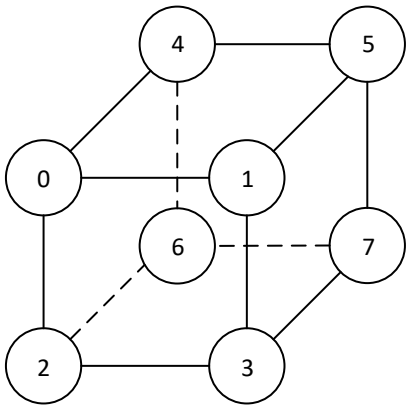
   Числа на дугах графа означают количество передаваемой информации.

2. Создаем матрицу смежности по данному графу, т.е. матрицу обмена информацией(МОИ) Zмои.

	1	2	3	4	5	6	7	8
1		20	40		35			
2	20			10	15	10		
3	40							28
4		10				7	6	
5	35	15					20	
6		10		7				12
7				6	20			
8			28			12		

   Для упрощения нулевые элементы не показаны.

3. Задана кубическая топология. Число процессоров не ограничено, но главное, чтобы это был куб из процессоров (в данном примере 2x2x2, т.к. вершин графа 8, соответственно 8 процессоров, в случае если вершин будет 9, то количество процессоров увеличится, так будет 3x3x3, т.е. 27 процессоров). Так как топология кубическая естественно всё измеряется в 3ех измерениях.



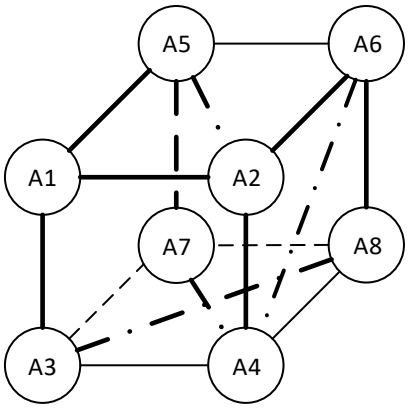
4. Создаем матрицу минимальных расстояний(ММР) топологии Zp(по примеру куб 2x2x2, соответственно ММР будет размерностью 8x8, если куб 3x3x3,то ММР 27x27 и т.д.).

	0	1	2	3	4	5	6	7
0	0	1	1	2	1	2	2	3
1	1	0	2	1	2	1	3	2
2	1	2	0	1	2	3	1	2
3	2	1	1	0	3	2	2	1
4	1	2	2	3	0	1	1	2
5	2	1	3	2	1	0	2	1
6	2	3	1	2	1	2	0	1
7	3	2	2	1	2	1	1	0

5. Для размещения графа в кубе, «накладываем» матрицу МОИ, т.е. матрицу смежности графа, на матрицу ММР(матрица расстояний в кубе, она остается неизменной для заданного куба). По примеру это выглядит так, мы разместили граф в кубе, это начальный и не оптимальный вариант:

	0	1	2	3	4	5	6	7
0	0	1	1	2	1	2	2	3
1	1	0	2	1	2	1	3	2
2	1	2	0	1	2	3	1	2
3	2	1	1	0	3	2	2	1
4	1	2	2	3	0	1	1	2
5	2	1	3	2	1	0	2	1
6	2	3	1	2	1	2	0	1
7	3	2	2	1	2	1	1	0

Куб с размещенным графом задач:



6. Выстраиваем информацию по убыванию (вектор B), а расстояния по возрастанию (вектор C) из полуматриц МОИ и ММР(то есть либо по правую сторону от нулей либо по левую, дабы не дублировались значения).
   По примеру соотвественно:
   B={40,35,28,20,20,15,12,10,10,7,6}
   C={1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3}

7. Далее вычисляем нижнюю оценку оптимального(идеального) размещения задач в кубической топологии.
   F=40*1+35*1+28*1+20*1+20*1+15*1+12*1+10*1+10*1+7*1+6*1=203
   Это то значение, к которому мы будем стремиться, чтобы разместить заданный граф в кубе оптимально, но самого этого значения мы не достигнем.
   По заданному примеру это число равно:
   F=40*1+35*1+28*2+20*1+20*1+15*2+12*1+10*1+10*1+7*2+6*2=259
   Из этого следует, что такое значение не соответствует оптимальному размещению. Далее пойдет объяснение того, как найти то оптимальное размещение.

Суть в том, чтобы мы меняли строки и столбцы в матрице МОИ таким образом, чтобы посчитанное новое значение F(при наложении измененной матрицы МОИ на неизменную ММР) было меньше предыдущего. То есть значения информации как можно больше совпадали с расстоянием единицы. Приоритет таков, чтобы самым большим числам(информации) доставалось значение расстояния единицы и по мере уменьшения самым маленьким значениям информации доставалось значение расстояния 2.

	1	2	3	4	5	6	7	8
1		20	40		35			
2	20			10	15	10		
3	40							28
4		10				7	6	
5	35	15					20	
6		10		7				12
7				6	20			
8			28			12		

	1	2	3	4	5	6	8	7
1		20	40		35			
2	20			10	15	10		
3	40						28	
4		10				7		6
5	35	15						20
6		10		7			12	
8			28			12		
7				6	20			

Пример: Старая МОИ 8↔7 Новая МОИ

Данной перестановкой мы уменьшили значение F с 259 до 257. Значение 257 мы сохраняем, далее мы меняем строки и столбцы либо перебором (т.е 1↔2, 1↔3, 1↔4, 1↔5, 1↔6, 1↔7, 1↔8, 2↔3, 2↔4 и т.п.) либо любым другим эффективным способом. После каждой перестановки надо посчитать значение F и сравнить его с предыдущим(если новое значение F меньше предыдущего, то новое значение F и соответственную перестановку сохраняем). Лучшую перестановку сохраняем и также сохраняем МОИ с этой перестановкой и назначаем её как основную(т.е. новая МОИ). Далее мы также делаем перестановки случайные или перебором или другим способом(и сохраняем лучший результат и «лучшую матрицу») до тех пор пока каждое новое значение F меньше предыдущего.

Если делать перестановки перебором, то надо сделать все перестановки, вычислить их значения F(занести их в какой-нибудь массив данных) и выбрать наименьшее значение из этого массива, потом сравнить с предыдущим значением F. Если оно окажется меньше, то сохраняем эту перестановку(например 2↔3) и заменяем предыдущее F на новое(которое появилось при лучшей перестановке, например 2↔3), создаем новую МОИ с учетом лучшей перестановки(например 2↔3) и далее начинаем новый перебор в новой МОИ(в дальнейшем старую МОИ заменяем на новую). Мы повторяем перебор, делаем новую МОИ с учетом лучшей перестановки и меньшим значением F чем предыдущее, до тех пор пока меньшее значение F с лучшей перестановкой массиве переборов не будет равна предыдущему сохраненному значению F.

1. Ввести M=$\|m_{ij}\|_{N\times N}$, $i=\overline{1,N}$, $j=\overline{1,N}$ $\forall m_{ij}=0$.

2. Ввести D=$\|d_{ij}\|_{N\times N}$, $i=\overline{1,N}$, $j=\overline{1,N}$.

3.1. Ввести M1= $\|m1_{ij}\|_{N\times N}$, $i=\overline{1,N}$, $j=\overline{1,N}$ $\forall m1_{ij}=0$.

3.2. Ввести M2= $\|m2_{ij}\|_{N\times N}$, $i=\overline{1,N}$, $j=\overline{1,N}$ $\forall m2_{ij}=0$.

3.3. Ввести M3= $\|m3_{ij}\|_{N\times N}$, $i=\overline{1,N}$, $j=\overline{1,N}$ $\forall m3_{ij}=0$.

4. Переписать $\forall m_{ij}$ в $\overline{M}=\left\|\overline{m}_{ij}^{z}\right\|$ так, что $m_{i,j}^{z_1} \geq m_{i,j}^{z_2} \Leftrightarrow z_1 < z_2$, где z_1 и z_2 порядковые номера элементов $\overline{M}$.

4. Переписать $\forall d_{ij}$ в $\overline{D}=\left\|\overline{d}_{ij}^{z}\right\|$ так, что $d_{i,j}^{z_1} \leq d_{i,j}^{z_2} \Leftrightarrow z_1 < z_2$, где z_1 и z_2 порядковые номера элементов $\overline{D}$.

6. Положить $T_{\inf} = \max\{m^z d^z\}$, где $z=\overline{1,|E|}$, $E \leq N^2 - N$.

7. Найти $T_н = \max\{m_{ij} \cdot d_{ij}\}$, $j = \overline{1,N}$, где N – число задач.

8. Вычислить $\eta_н = \frac{T_н}{T_{\inf}}$. Если $\eta < \eta_П$, то останов, иначе перейти к п.9.

9. Принять $T_0=T_н$.

10. Выполнить $M2 = M$: $\forall(l, j) \Rightarrow M2_{lj} = M_{lj}$.

11. Принять k=1.

12. Выбрать m_k из $\overline{M}$.

13. Найти $m2_{\alpha\beta}$ из M2 такой, что $m2_{\alpha\beta} = m_k$, и – соответствующий ему $d_{\alpha\beta}$ из D.

14. Если $d_{\alpha\beta}=1$, то $M2_{\alpha\beta} = -1$, $k=k+1$ и перейти к п.12, иначе п.15;

14. Принять i=1.

16. Если $i \le 64$, то перейти к п. 17, иначе – п.27.

17. Если $d_{\alpha i} < S$ и $d_{\alpha i} \neq 0$ и $m_{\alpha\beta} \cdot d_{\alpha\beta} > m_{\alpha i} \cdot d_{\alpha i}$ и $m_{\alpha\beta} \cdot d_{\alpha\beta} > m_{\alpha i} \cdot d_{\alpha\beta}$, то перейти к п.18, иначе $i=i+1$ и – п.16.

18. Для M выполнить операцию $i \leftrightarrow \beta$ и сформировать $M1$:

$\forall((l, j \neq i) \& (l, j \neq \beta)) \Rightarrow M1_{lj} = M_{lj}$, $\forall((l = i) \& (j \neq \beta)) \Rightarrow M1_{lj} = M_{\beta j}$,

$\forall((l = \beta) \& (j \neq i)) \Rightarrow M1_{lj} = M_{ij}$, $\forall((l \neq i) \& (j = \beta)) \Rightarrow M1_{lj} = M_{li}$,

$\forall((l \neq \beta) \& (j = i)) \Rightarrow M1_{lj} = M_{l\beta}$, $M1_{i\beta} = M_{\beta i}$, $M1_{\beta i} = M_{i\beta}$.

19. Вычислить T_1. Если $T_1 \le T_0$, то перейти к п.20, иначе $i = i+1$ и – п.16.

20. Вычислить $\eta = \frac{T_н}{T_1}$. Если $\eta < \eta_П$, то останов и выдача M1, иначе перейти к п.21.

21. Принять $M = M1$: $\forall(l, j) \Rightarrow M_{lj} = M1_{lj}$.

22. Принять $T_0=T_1$.

23. Принять $M2_{\alpha\beta} = -1$.

24. Для $M2$ выполнить операцию $i \leftrightarrow \beta$ и сформировать $M3$:

$\forall((l, j \neq i) \& (l, j \neq \beta)) \Rightarrow M3_{lj} = M2_{lj}$, $\forall((l = i) \& (j \neq \beta)) \Rightarrow M3_{lj} = M2_{\beta j}$,

$\forall((l = \beta) \& (j \neq i)) \Rightarrow M3_{lj} = M2_{ij}$, $\forall((l \neq i) \& (j = \beta)) \Rightarrow M3_{lj} = M2_{li}$,

$\forall((l \neq \beta) \& (j = i)) \Rightarrow M3_{lj} = M2_{l\beta}$, $M3_{i\beta} = M2_{\beta i}$, $M3_{\beta i} = M2_{i\beta}$.

24. Принять $M2 = M3$: $\forall(l, j) \Rightarrow M2_{lj} = M3_{lj}$.

26. Принять $k=k+1$ и перейти к п.28.

27. $M2_{\alpha\beta} = -1$, $k=k+1$.

28. Если $k < |\overline{M}|$, то останов и выдача M1, иначе перейти к п.11

ДАННЫЙ АЛГОРИТМ НЕСЕТ В СЕБЕ ОЗНАКОМИТЕЛЬНЫЙ ХАРАКТЕР ДЛЯ БОЛЬШЕГО ПОНИМАНИЯ ПЕРЕСТАНОВОК. ОН ПОХОЖ НА ТОТ, ЧТО ОПИСАН В ТЕКСТЕ ВЫШЕ. ПУНКТЫ ИЗ ЭТОГО АЛГОРИТМА МОГУТ БЫТЬ ПРИМЕНЕНЫ В ПРОГРАММЕ С АДАПТАЦИЕЙ ИХ ПОД КУБИЧЕСКУЮ ТОПОЛОГИЮ. В ПРИМЕРНОМ АЛГОРИТМЕ РАССМОТРЕННО РАЗМЕЩЕНИЕ В МАТРИЧНОЙ ТОПОЛОГИИ КЛАСТЕРНЫХ СИСТЕМ.