1 Исходные данные

Ориентированный граф работ без циклов, в котором дуги - зависимости по данным, а вершины - задания. Вершин n, дуг m
Вычислительная система, состоящая из p различных процессоров
Матрица C_{ij} длительности выполнения работ на процессорах, $i = 1 \dots n$, $j = 1 \dots m$
Матрица D_{kl} передач данных между процессорами, $k = 1 \dots p$, $l = 1 \dots p$, $D_{kk} = 0$

2 Требуется получить

Расписание заявок – упорядоченное множество $HP = \{j, s_j\}_{k=1}^{N}$, где k – порядковый номер j-й заявки, s_j – время выполнения k-й заявки в расписании HP. Корректность расписания определяется по следующим условиям:

1. В расписании не допустимы прерывания
2. Интервалы выполнения заявок не пересекаются
3. Каждая работа назначена на процессор
4. Любую работу обслуживает один процессор

3 Конфигурации задачи:

1. Задача с однородными процессорами (длительность выполнения задачи не зависит от того, на каком процессоре она выполняется) и дополнительными ограничениями на количество передач:
   (a) $CR = \frac{m_{ip}}{m}$
   (b) $CR2 = \frac{m_{2edg}}{m}$
2. Задача с однородными процессорами и контролем сбалансированности распределения работ
   (a) $BF = \left(\frac{a_{max} \cdot p}{n}\right) - 1$
3. Задача с неоднородными процессорами, но без дополнительных ограничений на расписание

4 Алгоритм

1. Сформировать множество вершин, у которых нет предшественников. Множество $D = (d_1, d_2 \dots d_i)$ где d_i – номер работы, доступной для добавления в расписание (т.е. у которой нет предшественников в исходном графе)

2. В случае, если в множестве D одна вершина – обозначим ее за d, в противном случае – создадим фиктивную вершину с нулевой длительностью, у которой все потомки будут из множества D, и обозначим ее за d

3. Зададим вектор k – вектор длин критических путей до вершин от d. При помощи алгоритма Дейкстры этот вектор заполняется значениями k_i, где i – номер вершины. Поскольку алгоритм Дейкстры работает со взвешенными графами, каждое ребро получает вес минимального времени работы на вычислительной системе вершины, из которой исходит

4. По жадному критерию выбора работы выбирается работа из множества D для размещения в расписании. Пусть выбранная работа – d_i

5. Производится пробное размещение работы d в расписании с учетом жадного критерия выбора места работы и дополнительных ограничений. В случае, если не получилось найти подходящее место для работы – запускается процедура ограниченного перебора с проверяемым критерием возможности добавления работы в расписание. Становится известно s – время старта работы и p – процессор, на котором работа выполняется. Выбор места выполнения работы происходит по системе допусков. Изначально места ранжируются по жадному критерию и берутся верхние $n\%$ (n – параметр алгоритма) списка. Оставшиеся места ранжируются по первому дополнительному ограничению, после чего снова берутся верхние $n\%$, и так далее по всем ограничениям. После прохода по всем ограничениям, из оставшегося списка берется место в соответствии с жадным критерием

6. Если s_i больше длины критического пути (с точностью до Δ, где Δ – параметр алгоритма), то вызывается процедура ограниченного перебора с проверяемым критерием $S = \sum s_k$, где s_k – времена начал всех перебираемых работ, в результате которой работа размещается в расписании. Если работу разместить не удалось – завершить алгоритм. Если s_i не превосходит длину критического пути(с точностью Δ), то работа размещается в расписании.

7. d_i удаляется из списка размещенных работ и в графе G удаляется соответствующая вершина и все дуги, исходящие из нее.

8. Обновляется множество D. Если D не пустое, то алгоритм переходит на пункт 4.

4.1 Жадный критерий выбора работы

- Максимальное количество потомков у работы

4.2 Жадный критерий выбора места работы в расписании

- Скорейшее завершение частично построенного расписания

4.3 Ограниченный перебор

После неудачной пробной постановки работы в расписание алгоритм создает набор $K = (k_1, k_2, \ldots, k_t)$, состоящий из t последних добавленных работ (t – параметр алгоритма). Далее, процедурой полного перебора пробуются различные расписания до тех пор, пока не получится расписание, удовлетворяющее критерию критичности пути до последней поставленной работы и удовлетворяющее дополнительным ограничениям.

4.4 Расчет времени начала работы

Для того, чтобы рассчитать время начала для конкретной работы на процессоре p требуется:

1. Вычислить вектор $PJ_{k=1}^{L}$, где L – количество предшественников у работы. Элементами этого вектора будут являться суммы вида $s_k + C_{kr} + D_{rj}$, где r – номер процессора, на котором размещен предшественник.
2. Максимумом этого вектора и будет являться первое доступное начало выполнения работы на данном процессоре. $n_j = \max PJ$