**Задача 4.1 [4]. Станки равной производительности**

*Определение* 1. Расписание называется до­пустимым, если в нем нет запаздывающих ра­бот. Работа называется допустимой, если она выполняется без запаздывания.

*Постановка задачи.* Имеется *m* независимых параллельных станков равной производи­тельности, работающих без прерываний, кото­рые выполняют *n* работ (*li* – длительность вы­полнения *i*-й работы, *i = *). Работы должны быть выполнены к директивным срокам *di*. Мо­менты запуска станков произвольны. Необхо­димо построить допустимое расписание, максимизирующее критерий:

*Критерий 1.*

 (1)

где *rj* – момент запуска станка *j*, *i*1 – номер станка, у которого момент запуска в опти­мальном расписании самый ранний (он является самым поздним для всех допустимых расписа­ний); *jl*,  – номер станка, у которого момент запуска следующий по величине после станков *jk*,  (он является самым позд­ним для всех допустимых расписаний с фикси­рованным ).

*Определение* 2. Допустимое расписание, у ко­торого моментами запуска станков являются , называется оптимальным по прямому лексикографическому порядку.

*Примечание*. Допустимое расписание называ­ется оптимальным по обратному лексикографи­ческому порядку, если для него выполняется:



*Признаки оптимальности допустимого ре­шения по векторному критерию (1).*

Обозначим через *I* множество всех работ {*i*: *i = *}.

Очевидно, выполняются неравенства *di* – *li* ≥  0, *i = *. Перенумеруем работы по не убыва­нию чисел *di* – *li*.

В статье [4] сформулированы и доказаны достаточные признаки оптимальности №1 и №2, которые позволили сконструировать полиномиальную составляющую ПДС-алгоритма для рассматри­ваемой задачи.

***Признак оптимальности №1 допустимого расписания.***

Пусть выполняются неравенства: *d*1 – *l*1 < *d*2 – *l*2 < … <*dn* – *ln* .

п. 1. На первый станок первой назначаем ра­боту с индексом 1. *r*1 = *d*1 – *l*1. Числу *r*1 соответст­вует максимально возможное значение  в (1). Пусть выполняются следующие неравенства: *di* + *lk* > *dk*, *i = *, *k* = **. Тогда назначаем на *k*-й станок, *k* = **, первой работу с индексом *k* и моментом запуска станка *rk* = *dk* – *lk*.

**Утверждение 1 [4].** Произвольное допусти­мое расписание, для которого выполнен п. 1, является оптимальным по критерию (1).

***Признак оптимальности №2 допустимого расписания.***

Пусть выполняются неравенства:

*d*1 – *l*1 < *d*2 – *l*2 < … <*dn* – *ln*  (2)

п. 2. На первый станок первой назначается работа с индексом 1. *r*1 = *d*1 – *l*1 (*d*1 – *l*1 *–* этомаксимально большое возможное значение  в (1)). Пусть *k*2 – 1 – максимальное натуральное число, для которого выполняются неравенства:

. (3)

Тогда на первый станок последовательно назначаются работы с индексами 1, 2, …, *k*2 – 1. Работа с индексом *k*2 назначается первой на второй станок, . Если неравенство



не выполняется, тогда на третий станок первой назначается работа с индексом *k*2 + 1 в момент времени . В этом случае *k*3 = *k*2 + 1. В противном случае должны выпол­няться неравенства:

, (4)

. (5)

Тогда работа с индексом *k*2 + 1 назначается на станок, которому соответствует минимум в (4). Если неравенство (4) выполнено, а неравенство (5) нарушается, то признак оптимальности №2 допустимого расписания для данной индивиду­альной задачи нарушен. Аналогично последова­тельно назначаются на первый либо второй станок работы с индексами  (*k*3 – макси­мально возможное натуральное число). При этом, если текущая работа может быть на­значена на станок с меньшим моментом начала обслуживания, то назначение ее на станок с большим моментом времени начала обслужива­ния должен приводить к нарушению директив­ного срока (аналог выполнения неравенств (4), (5)). Работа с индексом *k*3 первой назначается на третий станок в момент времени .

Аналогично происходит дальнейшее после­довательное назначение работ с индексами *k*3+*i* на станки. Необходимым условием выполне­ния признака оптимальности №2 является тре­бование, что распределяемая работа может быть назначена только на один станок (с минималь­ным временем освобождения) из текущего мно­жества станков, на которые производится на­значение работ. Если ни на один станок из те­кущего множества станков работа не может быть назначена, она назначается первой на сле­дующий станок в момент времени, равный ди­рективному сроку этой работы минус ее дли­тельность. Распределение работ заканчивается либо когда все работа распределены на *l* станков (*l* < *m*), либо назначением на *m*-й станок первой *km*-й работы . Распределе­ние работ в соответствии с п. 2 завершено.

Пусть *I*1 – множество работ, которое содер­жит все работы с индексами из множества {} либо {}, если работы распредели­лись на *l* станках (*l* < *m*). Потребуем выполне­ния следующего условия. Перенумеруем все работы из множества *I*1 в порядке их назначе­ния на станки. Тогда для каждой работы с индексом *i* () должно выполняться

*di* – *Ci* < *lp*, , (6)

где *Ci* – момент окончания выполнения *i*-й ра­боты. Тогда имеет место утверждение:

**Утверждение 2 [4].** Произвольное допусти­мое расписание, для которого выполнен п. 1 либо п. 2 и условие (6), является оптимальным по критерию (1), т.е.

≽ (7)

в соответствии с предпорядком – лексикогра­фическим порядком, где  – вектор мо­ментов запуска станков произвольного допус­тимого расписания, компоненты которого рас­положены в прямом лексикографическом по­рядке, т.е. первая компонента наименьшая, вто­рая – следующая по величине и т.д., а  соответствует допустимому расписанию, у ко­торого выполнен п. 1 либо п. 2 и условие (6) (компоненты вектора  по построению рас­положены в соответствии с прямым лексикографическим по­рядком).

Условие (7) соблюдается, если , либо если  и , либо  и , но , и т.д., либо .

Аналогично (7) по прямому лексикографическому по­рядку сравниваются два произвольных вектора.

*Примечание.* Если в соответствии с п. 2 за­груженными оказались *l* станков (*l* < *m*), то в (7) *rl+i*, , формально принимают значе­ния +∞.

*Примечание*. Если все работы распреде­лены по *l* станкам, *l* < *m*, то построено опти­мальное расписание по критерию (1).

***Признак оптимальности №3 допустимого расписания.***

*Анализ признаков оптимальности №1, №2 и возможности их улучшения.*

Очевидно, что признак оптимальности №1 является самым слабым, поскольку он может сработать только в тех случаев, когда первые заданий попарно пересекаются между собой на оси времени (если сдвинуть их все максимально поздно к своим директивным срокам).

Признак №2 расширяет предыдущий, но так же имеет свои «слабые места»:

1) Неравенства (4), (5) предполагают, что на каждой итерации работы алгоритма текущее задание можно назначить не более чем на один уже выделенный до этой итерации станок. А если директивные сроки заданий и их длительности такие, что задания являются «разреженными» на оси времени и/или количество станков достаточно большое, то данный случай может проявляться часто.

2) Условие (6) предполагает, что текущее назначаемое задание не должно быть возможным назначить на какой-либо станок не последним, то есть вставить между уже назначенными заданиями на данный станок. Аналогично предыдущему пункту, если задания «разрежены», существует высокая вероятность, что признак №2 не сработает ни для одного варианта назначения при таких входящих данных.

3) Рассматривается только случай, когда нет заданий с одинаковой разностью .

Устранение недостатков:

1) Алгоритм №3 может построить оптимальное начальное расписание и после возникновения ситуации, когда текущее задание может быть назначено более чем на один станок. В таком случае назначение заданий продолжается по эвристическому принципу. Разумеется, что если на какой-либо итерации эвристического назначения оказывается, что для задания не существует допустимых станков, мы не можем утверждать, что при назначении задания на свободный станок расписание останется оптимальным. Для этого вводится процедура проверки слабого условия выделения станка, которая за полиномиальное время проверяет, точно ли при выделении свободного станка расписание останется оптимальным. Но если данная процедура дала отрицательный результат, то есть допустимые варианты назначения на уже выделенные станки могут существовать, это не означает, что такие варианты назначения точно существуют (то есть, данная процедура теоретически может быть расширена). В таком случае рассматривается возможность выполнения процедуры строгого перебора возможных вариантов назначений. Условие запуска – количество участвующих в переборе заданий не превышает некоторой ограничительной постоянной . Таким образом, при больших значениях время выполнения процедуры может быть значительным, но однозначно является полиномиальной зависимостью от числа заданий , поскольку количество перебираемых вариантов ограничено величиной **, где – постоянная величина. Результат выполнения данной процедуры не только показывает возможность оптимального выделения свободного станка, но и дает информацию о всех допустимых вариантах назначения всех заданий до текущего включительно. Если же условие запуска процедуры строгого перебора не сработало, то есть количество участвующих в переборе заданий превысило , только в этом случае алгоритм завершает свою работу не построив оптимальное начальное расписание.**

2) Алгоритм №3 построения оптимального начального расписания построен таким образом, что на каждой итерации назначенные задания сохраняют порядок по не убыванию директивных сроков на каждом из станков. Для очередного задания проверяется возможность его назначения на каждый из станков на свою позицию по не убыванию директивных сроков заданий на данном станке. Если же задание невозможно допустимо назначить на свою позицию, не сдвигая задания, которые идут перед ним на данном станке, это означает, что допустимого назначения вместе с текущим заданием не существует, если не изменять момент запуска данного станка.

3) Признак оптимальности №3 исключает недостаток №3: далее будет рассмотрен случай, когда условие (2) не выполняется.

***Алгоритм №3 построения оптимального начального расписания.***

**Далее будем считать, что (иначе мы имеем тривиальный случай).**

**Пусть – некоторая постоянная, которая будет ограничивать глубину перебора для сохранения полиномиальной зависимости времени работы алгоритма от числа .**

**Перенумеруем задания по не убыванию чисел .**

**Для начала, предположим, что .**

**Работа алгоритма состоит из набора однотипных итераций, на каждой из которых назначается очередное задание на один из уже выделенных станков, если это возможно так, чтобы не изменилось время запуска данного станка, иначе проверяются условия оптимальности начального расписания в случае выделения свободного станка.**

**Итерации продолжаются до тех пор, пока не выполнится одно из условий:**

**1) на каждый из станков назначено хотя бы одно задание;**

**2) все задания назначены на выполнение;**

**3) текущее задание не получается назначить ни на один из выделенных станков так, чтобы начальное расписание оставалось допустимым, при этом ни одно из условий выделения нового станка не выполняется.**

**На первой итерации назначается задание с индексом на станок с индексом с началом выполнения в момент времени .**

**Пусть после итерации :**

**– множество индексов всех заданий, назначенных на выполнение; – множество индексов выделенных станков; – множество индексов заданий, назначенных на выполнение на станок с индексом , тогда = , , .**

**После каждой итерации будет выполняться:**

**.**

**Обозначим множество индексов выделенных станков, которые являются допустимыми для текущего задания на итерации , . Множество на текущей итерации определяется следующим образом.**

**Для каждого станка с индексом из множества пытаемся вставить текущее задание на свое место по не убыванию директивных сроков. При этом задания проверяемого станка, которые расположены позже этого места, сдвигаются ближе к своим директивным срокам на величину .**

**Пусть , где – множество индексов всех сдвинутых заданий проверяемого станка, – момент окончания выполнения задания с индексом .**

**Тогда:**

**где, , если (или на итерации сработало условие возобновления однозначности, которое будет описано далее), иначе .**

**Пусть – множество «неоднозначных» заданий на итерации :**

**На итерации выполняем следующую процедуру:**

**1. Проверяем условия 1), 2) окончания работы алгоритма. Если одно из них сработало – начальное расписание построено, алгоритм завершает свою работу. В противном случае переходим к пункту 2.**

**2. Если , назначаем текущее задание на один из станков (любой), которому соответствует минимум: , на позицию по не убыванию директивных сроков, переходим к пункту 3. В противном случае переходим к пункту 4.**

**3. Если , запускаем процедуру строгого перебора допустимых вариантов назначений. Если процедура показала, что существует только один вариант назначений (текущий) – условие возобновления однозначности выполняется для текущей итерации, очищаем множества . Переходим к следующей итерации.**

**4. Если , выделяем очередной станок, назначаем на него текущее задание с началом выполнения в момент , переходим к следующей итерации. В противном случае переходим к пункту 5.**

**5. Проверяем слабое условие выделения свободного станка. Если условие выполняется, выделяем очередной станок, назначаем на него задание с началом выполнения в момент времени , удаляем из множества , переходим к следующей итерации. Иначе переходим к пункту 6.**

**6. Если – начальное расписание для текущего набора входных данных построить не удалось, алгоритм завершает свою работу. В противном случае выполняем процедуру строгого перебора вариантов назначений, по результату выбираем действие:**

**1) если нет ни одного допустимого варианта назначения – выделяем очередной станок, назначаем на него задание с началом выполнения в момент времени , удаляем из множества ;**

**2) если существует один и только один вариант назначения – (пере-)назначаем все задания с индексами из множества (включая текущее) согласно определенному допустимому варианту, условие возобновления однозначности выполняется для текущей итерации, очищаем множества ;**

**3) иначе из определенных допустимых вариантов назначения выбираем произвольный, (пере-)назначаем все задания с индексами из множества (включая текущее) согласно выбранному варианту;**

**, переходим к следующей итерации.**

***Процедура проверки слабого условия выделения станка.***

**Пусть – множество индексов станков, пройденных для группы заданий, порожденных заданием , изначально .**

**0. Создаем копию текущего расписания без заданий с индексами из множества . Задания, которые находятся позднее хотя бы одного образовавшегося пробела в пределах своих станков, сдвигаются ближе к временам начала выполнения станков так, чтобы не было интервалов простоя. Создаем фиктивные задания – копии заданий с индексами из множества .**

**1. Если множество фиктивных заданий не пустое, выбираем из него задание с минимальным директивным сроком, переходим к пункту 2. В противном случае условие выделения станка не выполняется, процедура оканчивает свою работу.**

**2. Если множество не пустое, выбираем из него произвольный индекс , переходим к пункту 3. В противном случае условие выделения станка выполняется, выходим из процедуры.**

**3. Находим момент времени (позицию) для задания по не убыванию директивных сроков на станке , вычисляем , где – множество индексов всех заданий станка , которые расположены после момента времени , – момент окончания выполнения задания с индексом . Переходим к 4.**

**4. Если , назначаем задание на станок с началом в момент времени , задания с индексами из множества сдвигаются ближе к своим директивным срокам на величину . Иначе, если , разбиваем задание на два фиктивных задания с директивными сроками и длительностями и , назначаем задание длительности на станок аналогичным образом. Пополняем множество индексом , переходим к пункту 1.**

***Процедура строгого перебора возможных вариантов назначений.***

**Далее вариантом распределения заданий из множества по станкам из множества будем считать вектор , где – индекс станка, соответствующего заданию . Пусть – множество всех возможных вариантов распределений заданий с индексами из множества по станкам. Количество таких вариантов:**

**Пусть – множество пройденных вариантов распределений (изначально ), – множество допустимых вариантов назначений, результат работы процедуры.**

**0. Создаем копию текущего расписания без заданий с индексами из множества . Задания, которые находятся позднее хотя бы одного образовавшегося пробела в пределах своих станков, сдвигаются ближе к временам начала выполнения станков так, чтобы не было интервалов простоя.**

**1. Если множество не пустое, выбираем из него произвольный вариант распределения , переходим к пункту 2. Иначе процедура оканчивает свою работу.**

**2. Пытаемся назначить все задания на станки согласно варианту распределения . При этом каждое очередное задание пытаемся вставить на свое место по не убыванию директивных сроков на станке . Задания станка, которые расположены позже этого места, сдвигаются ближе к своим директивным срокам на величину . Если такое назначение является допустимым, добавляем данный вариант назначения в результирующее множество . Множество пополняется вариантом распределения , переходим к пункту 1.**

***Примечание.* Так как величина сверху ограничена числом , рекомендуется выполнять данную процедуру при ограниченном количестве элементов во множестве . Иначе верхняя оценка времени работы алгоритма будет экспоненциальной зависимостью от числа заданий . Для этого алгоритм №3 построения оптимального начального расписания определяет ограничивающую постоянную .**

***Случай невыполнения условия (2).***

**Теперь рассмотрим случай, когда . Пусть – множество индексов заданий таких, что и , пусть – количество элементов множества .**

**Так как задания перенумерованы по не убыванию чисел , то задания множества идут подряд по ходу итераций процедуры. Если все задания множества имеют одинаковые длительности, процедура построения начального оптимального расписания остается без изменений. Если для самого короткого задания из множества нет допустимых станков (), процедура остается без изменений.**

**В противном случае перенумеровываем задания по не возрастанию длительностей. Если на протяжении всех итераций не было выделено ни одного нового станка, процедура остается без изменений. В противном случае:**

**1) , где ;**

**на каждой итерации текущее задание добавляем во множество , при этом условия возобновления однозначности не рассматриваются;**

**3) при выделении станка , во все множества добавляем .**

**Эти пункты выполняются на всех итерациях . Таким образом, мы фиксируем времена запусков выделенных станков, но оставляем возможность перестановки заданий множества между собой.**

***Экспериментальный анализ признаков оптимальности №1, 2, 3 в зависимости от входных данных.***

**Предположим, что длительности заданий являются случайной величиной , имеющей гамма-распределение , а интервалы между директивными сроками – случайная величина , которая распределена по экспоненциальному закону .**

**Постоянные параметры:**

**,**

**,**

**,**

**,**

**.**

**Переменные параметры:**

**– коэффициент масштаба в экспоненциальном распределении интервалов.**

**Количество экспериментов на одно значение : 300.**

**Рисунок 1 – Графическое сравнение эффективности алгоритмов**

**На самом деле, законы распределения не играют большой роли. Показатели эффективности признаков оптимальности сильно зависят от величины (при достаточно больших и ), что будет показано далее.**