**Признак оптимальности №3 допустимого расписания.**

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

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

**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 объектов, если это возможно так, чтобы не изменилось время запуска данного объекта, иначе проверяются условия оптимальности начального расписания в случае выделения свободного объекта.**

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

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

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

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

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

**Пусть после итерации :**

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

**После каждой итерации будет выполняться:**

**.**

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

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

**Пусть , где – множество индексов всех сдвинутых заданий проверяемого объекта, – 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 объектов (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 объект, 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 объекта. Если условие выполняется, выделяем очередной объект, 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 назначений – выделяем очередной объект, назначаем на него задание с началом выполнения в момент времени , удаляем из множества ;**

**2) если существует один и только один вариант назначений – (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 назначений выбираем произвольный, (пере-)назначаем все задания с индексами из множества (включая текущее) согласно выбранному варианту;**

**, переходим к следующей итерации.**

***Процедура проверки слабого условия выделения объекта.***

**Пусть – множество индексов объектов, 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 объектов, сдвигаются ближе к временам начала выполнения объектов 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 объекта не выполняется, процедура оканчивает свою работу.**

**2. Если множество не пустое, выбираем из него произвольный индекс . В противном случае условие выделения объекта выполняется, выходим из процедуры.**

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

**3. Если , назначаем задание на объект 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 объект аналогичным образом. Пополняем множество индексом , переходим к пункту 1.**

***Процедура строгого перебора допустимых вариантов назначений.***

**Далее вариантом распределения заданий из множества по объектам из множества будем считать вектор , где – индекс объекта, соответствующего заданию . Пусть – множество всех возможных вариантов распределений заданий с индексами из множества по объектам. 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 объектов, сдвигаются ближе к временам начала выполнения объектов 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 объекты согласно варианту распределения . При этом каждое очередное задание пытаемся вставить на свое место по не убыванию директивных сроков на объекте . Задания объекта, 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 . В противном случае пополняем множество 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 сложности алгоритма будет экспоненциальной функцией зависимости от числа заданий . Для этого алгоритм №3 построения оптимального начального расписания определяет ограничивающую постоянную .**

**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 объектов (), процедура остается без изменений.**

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

**1) , где ;**

**на каждой итерации текущее задание добавляем во множество ;**

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

**Таким образом, мы фиксируем времена запусков выделенных объектов, но оставляем возможность перестановки заданий множества между собой.**

**Экспериментальный анализ эффективности алгоритмов.**

**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.**