Министерство образования Республики Беларусь

Учреждение образования

"Гомельский государственный университет им.Ф. Скорины"

Математический факультет

Кафедра МПУ

Реферат

**Метод приоритетов для задач разработки расписаний**

Исполнитель:

Студентка группы М-52

Ларченко А.Ю.

Научный руководитель:

Канд. физ-мат. наук, доцент

Звенцова Т.Е.

Гомель 2004

Содержание

Введение

1. О характере задачи

2. Можно ли её решить полным перебором

3. Множество D

4. Прогноз тупика

Заключение

Литература

## Введение

Данная работа посвящена проблеме разработки математической модели сложной задачи. Проблема необъятна, существующие методы на мой взгляд настолько общи, что в них мало смысла.

Поэтому я не буду заниматься изложением общих мест, а просто приведу пример такой разработки, достаточно сложный, чтобы он был интересен и достаточно понятный.

Конечно, описанная ниже модель, ни в коем случае не претендует на полноту и точность, это всего лишь (я надеюсь удачный) демонстрационный пример.

Я попробую разобрать очень популярную задачу, решить которую пытались и ныне пытаются очень многие программисты. Я имею ввиду задачу составления расписаний.

Конечно, это целый класс задач, но мы далее будем говорить только об одном представителе этого класса - задаче составления расписания учебных занятий. Однако этот представитель очень ярок и нам его будет достаточно.

## 1. О характере задачи

Грубо говоря существует два крайних типа задач. Первые это хорошие задачи для которых можно получить красивое аналитическое решение. То есть решение может выражаться каким-то компактным быстро считаемым утверждением, например формулой.

Элементарный пример аналитического решения - это решение квадратного уравнения. Сложность получаемых формул во внимание не берём. Кубическое уравнение имеет более сложное решение, но принципиально оно ничем не отличается от решения квадратного, это так же формула и не более того.

Второй тип задач - это плохие задачи, для которых необходим полный перебор. Вот пример такой плохой задачи "Найти самого высокого китайца". В этой задаче, как ни крутись, а придётся перебирать всех китайцев.

Прежде чем браться за задачу необходимо выяснить плохая это задача или хорошая. Потратим две минуты на анализ. Решение хорошей задачи заканчивается формулой. Формула это внешнее проявление внутренней закономерности присущей данным задачи.

А что такое закономерность? В самом общем виде - закономерность это связи между данными, какие-то ограничения на них. В задаче о расписании исходные данные могут быть в сущности какими угодно и они между собой никак не связаны.

Если мы к примеру знаем, что у Ивана Ивановича есть уроки математики в 10 классе, то это не даёт нам никакой информации о уроках русского в этом ж классе.

Поэтому мы вряд ли в праве ожидать закономерности, и поэтому задача теории расписаний это плохая задача.

## 2. Можно ли её решить полным перебором

Чтобы ответить на поставленный вопрос необходимо оценить количество выполняемых действий. Попробуем сделать это.

Для начала сформулируем задачу более точно.

Расписание это сетка уроков, по которой распределены занятия. Ячейки этой сетки будем называть в дальнейшем вакансиями, а занятия пусть оставят за собой свое название.

Предположим для упрощения, что количество вакансий равно количеству занятий и запишем какую-нибудь простейшую структуру данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | Понедельник | Вторник | Среда |
| Первый урок | 1 | 2 | 3 |
| Второй урок | 4 | 5 | 6 |

И пусть занятий только шесть. Пустые клетки это вакансии. Мы их пронумеровали, чтобы увидеть простой факт: хотя сетка вакансий и прямоугольная вакансии можно выстроить в простой ряд. Пронумеруем также и занятия А1, А2, А3, А4, А5, А6. Тогда задача поиска необходимого варианта расписания заключается в получении всех перестановок из 6 элементов. Известно же, что из N элементов можно получить N! = 1\*2\*3\*4…. \*N перестановок, то есть в нашей задаче 6! =1\*2\*3\*4\*5\*6 = 720. А для реального набора данных, например в 50 занятий число перестановок вообще получается астрономическим. Кроме того, необходимо помнить, что количество вакансий в реальных задачах больше количества занятий, а стало быть даже не очень объёмная задача теории расписания требует ресурсов суперкомпьютера.

*Небольшое, но важное дополнение.*

Почему нельзя взять первый попавшийся вариант? А потому, что на расписание накладывается ряд условий выполнение которых невозможно при произвольном варианте (например отсутствие дырок в расписании класса). Этих условий обычно очень много, и они резко сокращают количество допустимых вариантов. Фактически их так мало, что вероятность наткнуться на допустимый вариант в самом начале перебора практически равна нулю.

*Что же делать?*

Надеюсь, выше я достаточно ясно показал безнадежность нашего положения. Лобовая атака на задачу ничего не даст. Поэтому единственный выход - это изменить отношение к задаче. Например, мы можем отказаться от намерения получить гарантированно идеальное решение за короткое время. А давайте сформулируем наши намерения несколько иначе.

Теперь мы желаем просто максимально увеличить вероятность обнаружения достаточно хорошего варианта за ограниченное время. Мы смягчили свои запросы, и теперь можем рассчитывать на успех. Но сначала опишем задачу в более строгих терминах.

***Обозначения:***

**А -** Множество занятий

**а -** Занятие

**В -** множество вакансий

**в -** Вакансия

(**а, в) -** допустимая пара расписания, то есть пара не противоречащая требованиям налагаемым на расписание.

Вполне возможно, что для элемента **а** существует несколько допустимых пар расписания. А если это возможно для элемента **а,** то следовательно возможно и для элемента **в.** Таким образом можно ввести ещё два важных понятия:

**Ва -** множество все элементов **в** которые могут участвовать в допустимых парах расписания с элементом **а.**

**Ав -** множество все элементов **а** которые могут участвовать в допустимых парах расписания с элементом **в.**

Тогда расписанием назовём такое множество допустимых пар расписания в котором каждый элемента множества **А** присутствует ровно один раз.

Таким образом, расписание это элемент множества всех множеств допустимых пар. А составление расписания тогда математически сводится к поиску нужного элемента среди уже упомянутого **множества всех множеств допустимых пар (обозначим его как D).**

## 3. Множество D

На первый взгляд оно устроено беспорядочно. Однако это не так: возьмём какой-либо элемент этого множества **d.** Он представляет собой множество допустимых пар. Совершенно очевидно, что для данного элемента существует (и быть может не один) элемент **d'** отличающийся от **d** на одну пару и при этом **d>d'.** скажем тогда, что элементы **d** и **d'** связаны между собой отношением следования **d → d'.** Очевидно, что каждый элемент множества **D** связан отношением хотя бы с одним элементом. Если теперь, мы расположим элементы множества **D** на плоскости и те элементы которые находятся между собой в отношении следования соединим стрелками, то получим связный ориентированный граф. Это для тех кто знает, что такое связный ориентированный граф. Если кто не знает, то пусть не расстраивается, для нашей задачи не важно как это называется, важно представить себе эту картинку. А выглядит она примерно так.

Тогда процесс поиска элемента **d** являющегося расписанием есть ничто иное как путь вдоль стрелок ведущий к искомому элементу.

В общем это и есть модель. Мы свели поиск расписания к поиску пути на ориентированном графе. А ориентированный граф это структура, о которой в математике известно довольно много и теперь мы можем обрушить на задачу всю мощь теории графов. Но давайте предположим, что большинство из нас оной теорией не владеют, и продолжим поиск решения.

Но кое что из модели графа мы возьмём. Заметим, что каждый путь на графе обязательно заканчивается узлом, из которого не выходит ни одна ветка. То есть из этих узлов продолжать поиск расписания нельзя, а это означает, что имеет место одна из двух ситуаций:

Расписание уже составлено.

Расписание не составлено, но для некоторых элементов **а** нет ни одного элемента **в.** Будем называть дальше эту ситуацию тупиком.

Мы сможем ускорить процесс поиска расписания, если мы научимся определять тупиковый путь или нет не проходя его, иначе говоря мы должны научится делать

## 4. Прогноз тупика

В начале пути по графу каждый элемент **а** имеет непустую область определения **Ва** иначе процесс поиска расписания можно было бы и не начинать. Построение очередной пары расписания (переход в следующий узел графа) означает уменьшение множества **В** на одну вакансию и уменьшение областей определения некоторых элементов множества **А**. Предположим на некотором шаге у **а1** область определения состоит из 10 элементов **в,** а у **а2** область определения состоит из одного элемента **в.**

Если на следующем шаге область определения **а2** уменьшится на 1 то весь процесс зайдёт в тупик. То есть можно сформулировать очевидное утверждение: Наибольшую угрозу завести процесс в тупик представляют те элементы **а** у которых область определения наименьшая. А отсюда возникает хорошая и совершенно очевидная идея.

*Для того, чтобы минимизировать риск возникновения тупика необходимо на каждом шаге построения расписания выбирать такой элемент* ***а*** *у которого область определения наименьшая.*

Эта идея говорит о том, как выбирать элемент **а** для очередного действия по составления расписания, но ещё остаётся проблема как выбирать **в** в пару элементу **а.** Если область определения **Ва** состоит из одного элемента, то такой проблемы нет, но скорее всего она будет содержатьнесколько вакансий. Вопрос, какую из них выбрать. Проведём небольшое рассуждение.

Предположим, что в области **Ва** содержится два элемента **в1** и **в2** при этом **Ав1 (**область определения **в1**) содержит два элемента **а** и **Ав2** содержит пять элементов **а.** Это означает, что если мы возьмём для очередной пары расписания элемент **в1** то мы уменьшим на 1 область определения у двух элементов **а,** если же мы возьмём **в2** то мы уменьшим область определения у 5 элементов **а.** Думаю, критерий выбора **в** уже понятен и осталось записать общий алгоритм разработки расписания.

Рассчитать все области определения **Ав**

Рассчитать все области определения **Ва**

Пока **А** не пусто делать

Начало

Определить очередной **а (**элемент множества **А** с наименьшей областью определения)

В области определения **а** определить очередной **в (**элемент множества **В** с наименьшей областью определения)

Составить очередную пару расписания (очередной **а,** очередной **в**)

Вычеркнуть "очередной **а**" из областей определения всех **в** которые входят в область определения "очередного **а"**

Вычеркнуть "очередной **в**" из областей определения всех **а** которые входят в область определения "очередного **в"**

Конец

Последние два пункта алгоритма необходимы для учета изменений областей определений, а они изменяются каждый раз когда из **А** и из **В** выбирается очередной элемент.

## Заключение

Вроде бы интуитивно ясно, что если построенный алгоритм будет работать, то работать он будет лучше, чем полный перебор, но его краткость (в отношении к сложности задачи) наводит на мысль, что работать он как раз и не будет.

Это только начало, всего лишь основная идея. Ниже я опишу проблемы которые необходимо решить, что бы алгоритм стал действительно рабочим.

**Первая проблема.** Алгоритм построен с целью раннего прогнозирования тупиковых ситуаций, но совершенно очевидно, что он не может полностью исключить тупиков, а следовательно нужно либо усовершенствование модели, чтобы сделать прогноз точнее, либо нужна специальная процедура выхода из тупика.

Общую идею, выходя из тупика даёт построенный выше ориентированный граф. Если мы зашли в тупик, то необходимо сделать несколько шагов в обратном направлении, против направления стрелок.

**Вторая проблема.** Алгоритм вроде бы стремится к решению, но из его текста из описания модели совершенно не ясно, что он гарантирует нахождение решения.

Поэтому после того, как к нему будет добавлена процедура выхода из тупика, необходимо провести доказательство, что он действительно найдёт существующее решение.

Для доказательства можно будет например показать, что алгоритм в худшем случае гарантирует полный перебор вариантов или пользуясь нашей моделью можно сказать, что алгоритм гарантирует полный обход графа построенного на множестве **D.**

**Третья проблема.** Все рассуждения приведённые выше исходят из того, чтоб области определения элементов **а** могут только уменьшаться на 1. Но это не так. Приведём пример.

Пусть к расписанию предъявлено следующее требование "У десятого А класса в расписании не должно быть дыр". И пусть на некотором шаге для этого класса был поставлен урок "Понедельник 2 урок каб.11". Это означает, что все вакансии "Понедельник 4 урок" будут запрещены так как это создаст дырку.

Однако, если окажется заполненной вакансия "понедельник 3 урок" то вакансия "понедельник 4 урок" опять станет доступной. Из этого следует, что область определения может изменяться скачкообразно, как в сторону уменьшения так и в сторону увеличения.

**Четвертая проблема.** Ясно, что требования к расписанию обладают разной степенью значимости. Некоторые из них обязательно должны быть выполнены, а некоторыми можно и пожертвовать.

Но эти свойства требований к расписанию в модели описанной выше вообще никак не учтены.

И последнее. Мы пользовались этой моделью для решения некоторых частных задач на составление расписаний и могу сказать, что метод работает качественно.

## Литература

1. Вострикова З.П. и др. "Программирование на языке "БЕЙСИК" для персональных ЭВМ". Машиностроение, 1993г.
2. Гохман А.В. и др. "Сборник задач по математической логике и алгебры множеств", издательство Саратовского Университета, 1969г.
3. Гусев В.В. Основы импульсной техники.М. Советское радио, 1975
4. Касаткин В.Н. "Информация, алгоритмы, ЭВМ", М. Просвещение, 1991г.
5. Машовцев В.А. Вступительные экзамены по информатике // Информатика. 1997, №13
6. Орлов В.А. О вступительных экзаменах по информатике // Информатика, 1997, №15
7. Яснева Г.Г. Логические основы ЭВМ // Информатика и образование, 1998, №2
8. Лыскова В.Ю., Ракитина Е.А. Логика в информатике, М. Информатика и образование 1999
9. Шауцкова Л.З. “Решение логических задач средствами алгебры логики”, газета Информатика 1999, №5.