УДК 004.02.21

АДАПТИВНЫЙ АЛГОРИТМ РАСПРЕДЕЛЕНИЯ ЗАКАЗОВ НА ОБСЛУЖИВАНИЕ АВТОМОБИЛЯМИ ТАКСИ

Д.М. Сонькин

Томский политехнический университет
E-mail: sonkin_d@sibmail.com

На основе известных алгоритмов нахождения максимального паросочетания разработан адаптивный алгоритм распределения заказов на обслуживание автомобилями такси. Предложенные способы адаптации алгоритма Куна и венгерского метода позволили сократить вычислительную сложность программных реализаций.

Ключевые слова:
Адаптивный алгоритм, распределение, паросочетание, автоматизация.

Key words:
Adaptive algorithm, distribution, matching, automation.

Введение

В настоящий момент каждый таксопарк решает задачу распределения заказов по-своему. Наиболее распространенным является подход, когда город делится на части (районы) и в каждой части (районе) организуется стоянка автомобилей такси. Водители отмечаются на ближайшей стоянке, организуя живую очередь. Поступивший заказ отдается первому водителю с ближайшей стоянки.

Такой подход обеспечивает «прозрачность» распределения заказов между автомобилями такси, а также упрощает задачу принятия решения. Однако говорить об эффективности, а тем более оптимальности такого распределения заказов не приходится. Разделение города на стоянки носит виртуальный характер – местоположение автомобиля внутри района неизвестно. Водители заинтересованы быть в центре, т. к. там выше вероятность получить заказ. Зачастую водитель отмечается на стоянке заранее, чтобы занять очередь. В результате большинство машин оказывается в очереди в центре, и отсутствие автомобилей в ряде других районов. При этом физически автомобиль может находиться в районе, где на стоянке никто не отмечен, а заказ имеется. Растут перепробеги автомобилей, увеличивая себестоимость заказа и время на его обслуживание. В случае, если к заказанному автомобилю предъявляются дополнительные требования (наличие кондиционера, логотипа, детского сидения и др.), диспетчер тратит время на поиск подходящего варианта. Также не учитывается фактор выполнения «дневного плана» – одни водители успевают его перевыполнить, а другие закрывают смену с недостачей.

Формализация и математическая постановка задачи оптимального распределения заказов между водителями такси

Задачу автоматизации процесса распределения заказов на обслуживание заказов клиентов такси между автомобилями таксопарка целесообразно представить в виде двудольного графа $G=(Z,A)$, где $Z=\{z_1,...,z_n\}$ – множество поступающих заказов и $A=\{a_1,...,a_m\}$ – множество автомобилей такси, готовых их выполнить, а $Y=\{y_{ij}\}$; $i=\overline{1,n}$; $j=\overline{1,m}$ – множество ребер, связывающих вершины из множества Z с вершинами множества A.

Следует отметить, что в общем случае каждый из заказов может быть принят и выполнен любым автомобилем такси. Это соответствует случаю, когда каждая вершина $z_i \subset Z$ может быть связана со всеми вершинами множества A, т. е. коэффициент инцидентности γ_i^z каждой вершины z_i находится в диапазоне $0 \le \gamma_i^z \le m$. Аналогично, каждый автомобиль такси может обслужить любой поступивший заказ, т. е. каждая вершина $a_j \subset A$ может быть связана со всеми вершинами множества Z, и соответственно коэффициент инцидентности γ_j^A каждой вершины a_i находится в диапазоне $0 \le \gamma_j^A \le n$.

Таким образом, при распределении заказов необходимо из всего множества возможных заказов $\gamma = \sum_{i=1}^{n} \gamma_i^Z$ или $\gamma = \sum_{j=1}^{m} \gamma_j^A$, выбрать такую совокупность паросочетаний (заказ-автомобиль), которая в наилучшей степени удовлетворяла бы заданному интегральному критерию эффективности K.

Каждому ребру из множества Y может быть поставлен в соответствие интегральный критерий эффективности $k_{ij} \subset K$. При этом $k_{ij}=f(k_{ij}^1,k_{ij}^2,...,k_{ij}^p)$, где p – число локальных весовых коэффициентов, таких как: удаленность от места назначения, тип автомобиля, процент выполнения дневного плана, тип клиента и др.

Таким образом, задача распределения заказов может быть сведена к нахождению оптимального паросочетания вершин множества Z с вершинами множества A так, чтобы каждой вершине z_i соответствовала только одна вершина a_j и наоборот.

На основе формализованного представления задачу оптимального распределения заказов целесообразно свести к классической задаче о назначениях с аддитивным критерием эффективности [5].

Необходимо распределить полученные заказы (множество Z) между автомобилями такси таким образом, что бы достичь максимальной эффективности F при их выполнении:

$$F = \sum_{i=1}^{n}\sum_{j=1}^{m} k_{i,j}x_{i,j} \to \max,$$

при ограничениях на количество:

- заказов, назначенных к исполнению водителям такси

$$\sum_{j=1}^{n} x_{i,j} \le 1,\ \ i = 1,2,...,m;$$

- водителей такси, получивших к исполнению заказ

$$\sum_{i=1}^{m} x_{i,j} \le 1,\ \ j = 1,2,...,n;$$

$$x_{i,j} = \begin{cases} 1, \text{ если водителю с номером } i \text{ будет назначен заказ с номером } j; \\ 2, \text{ в противном случае,} \end{cases}$$

где $k_{i,j}$ – интегральный критерий эффективности, учитывающий p весовых локальных коэффициентов для i-го водителя при выполнении j-го заказа; $x_{i,j}$ – элемент искомой матрицы назначений (матрицы неизвестных).

Разработка алгоритма оптимального назначения заказов автомобилям такси

Поиск максимального паросочетания в графе представляет собой классическую алгоритмическую задачу. Мы рассмотрим ее решение не для общего случая, а для графов частного вида – двудольных графов.

Из литературы [4–7] известны алгоритмы поиска оптимального паросочетания, к которому сводится задача распределения заказов между автомобилями такси. Рассмотрим наиболее известные из них: алгоритм Куна и венгерский метод.

Модернизация алгоритма Куна для определения максимального паросочетания «автомобиль – заказ»

Главная цель функционирования таксопарка, как коммерческой организации, является получение максимальной прибыли при обслуживании максимального количества клиентов. Однако, в ряде случаев, при обслуживании клиентов на первый план может выходить не критерий экономической эффективности, а критерий обслуживания максимального количества заявок клиентов. В таком случае интегральные критерии эффективности $k_{ij} \subset K$ во внимание не принимаются, и двудольный граф $G=(Z,A)$ перестает быть взвешенным. Для поиска максимального паросочетания в этом случае целесообразно воспользоваться алгоритмом Куна.

В теории графов принято цепью длины k именовать некоторый простой путь (т. е. не содержащий повторяющихся вершин или рёбер), содержащий k рёбер. Чередующейся цепью (в двудольном графе, относительно некоторого паросочетания) назовём цепь, в которой рёбра поочередно принадлежат/не принадлежат паросочетанию. Увеличивающей цепью (в двудольном графе, относительно некоторого паросочетания) назовём цепь, у которой начальная и конечная вершины не принадлежат паросочетанию.

Из теоремы Бержа [7] известно, что паросочетание является максимальным тогда и только тогда, когда не существует увеличивающих относительно него цепей.

Далее, заметим, что если найдена некоторая увеличивающая цепь, то с её помощью легко увеличить мощность паросочетания на 1. Пройдём вдоль этой цепи, затем каждое ребро поочерёдно будем добавлять/удалять из паросочетания (т. е. первое ребро, которое по определению не принадлежит паросочетанию, добавим в паросочетание, второе удалим, третье добавим и т. д.). Действительно, в результате этой операции мы увеличим мощность паросочетания на 1 (для этого достаточно заметить, что длина увеличивающей цепи всегда нечётна, а корректность вышеописанного преобразования очевидна).

Таким образом, мы получили каркас алгоритма построения максимального паросочетания: искать в графе увеличивающие цепи, пока они существуют, и увеличивать паросочетание вдоль них. Осталось детализировать способ нахождения увеличивающих цепей. Алгоритм Куна основан на поиске в глубину или в ширину, и выбирает каждый раз любую из найденных увеличивающих цепей.

Оценим вычислительную сложность алгоритма. Поскольку каждая увеличивающая цепь будет найдена за $O(n+m)$, а всего цепей понадобится найти не более $n/2$, то итоговая асимптотика алгоритма равна $O(n/2+nm)$, т. е. количество условных вычислительных операций составит $O(nm)$.

Рассмотрим подробнее алгоритм поиска увеличивающей цепи (пусть, для определённости, это поиск в глубину). Поиск начинает идти из вершины первой доли. Из первой доли во вторую он ходит только по рёбрам, не принадлежащим паросочетанию, а из второй доли в первую – наоборот, только по принадлежащим. С точки зрения реализации, поиск в глубину всегда находится в вершине первой доли, и он возвращает булево значение – найдена цепь или не найдена. Из текущей вершины V поиск в глубину пытается пойти по всем смежным рёбрам (кроме принадлежащего паросочетанию), и если он может пойти в вершину T_0 второй доли, не принадлежащей паросочетанию, то возвращает True и добавляет ребро (V,T_0) в паросочетание. Если же он пытается пойти в вершину T_0, уже принадлежащую паросочетанию, то он вызывает себя из вершины $M_1[T_0]$ (соседа T_0 в паросочетании), и если цепь была найдена, то добавляет ребро (V,T_0) в паросочетание.

Модифицируем описанный выше алгоритм Куна следующим образом. До основного цикла алгоритма найдём каким-нибудь простым (жадным) алгоритмом произвольное паросочетание, и лишь затем будем выполнять алгоритм Куна, который будет улучшать это паросочетание. В результате алгоритм будет работать заметно быстрее на случайных графах – потому что в большинстве графов можно легко набрать паросочетание достаточно большого веса.

Например, можно просто перебрать все вершины первой доли, и для каждой из них найти произвольное ребро, которое можно добавить в паросочетание, и добавить его. Даже такая простая эвристика способна ускорить алгоритм Куна в несколько раз.

Следует обратить внимание на то, что алгоритм придётся немного модифицировать: поскольку предполагается, что текущая вершина ещё не входит в паросочетание, то нужно добавить соответствующую проверку.

Поиск оптимального распределения заказов венгерским методом

Значительно более сложной представляется задача о паросочетании, в котором задан граф $G=(Z,A,K)$ и каждому ребру $Y=\{y_{ij}\}$; $i=\overline{1,n}$; $j=\overline{1,m}$ – сопоставлено число k_{ij}, называемое весом ребра y_{ij}. Предлагается найти паросочетание в G с наибольшей возможной суммой весов. Очевидно, что задача о невзвешенном паросочетании, которую мы рассматривали выше (называемая иногда задачей о мощности паросочетания), – это частный случай задачи о взвешенном паросочетании: достаточно положить $k_{ij}=1$ для всех $y_{ij}\in Y$.

Нетрудно увидеть, что в формулировке задачи о взвешенном паросочетании можно убрать граф G, приняв соглашение, что этот граф всегда полный, и полагая веса всех ребер, которых нет в G, равными нулю. Кроме того, всегда можно считать, что число вершин четно – в противном случае можно добавить новую вершину и положить веса всех ребер, инцидентных этой вершине, равными нулю. Если рассматривается двудольный случай этой задачи, так же можно считать, что граф представляет собой полный двудольный граф с двумя множествами вершин одинаковой мощности. Отметим также, что оптимальные решения обязательно будут полными паросочетаниями (т. к. $k_{ij}\geq 0$), и, следовательно, можно формулировать эти задачи как задачи минимизации, рассматривая стоимости $c_{ij}=W-k_{ij}$, где W больше, чем все k_{ij}. Далее будем рассматривать именно такой вариант задачи

Рассмотрим алгоритм решения задачи, если требуется минимизировать произвольный показатель (например, пробег автомобилей). В противном случае (в случае задачи максимизации эффективности) выберем число W, большее любого элемента матрицы K, и составим новую матрицу K', элементы которой соответственно будут равны: $k'_{ij}=W-k_{ij}$.

Если матрица K не квадратная, дополняем ее нужным числом нулевых рядов. (Под рядом будем понимать строку или столбец).

Этап 1. Приводим матрицу следующим образом: уменьшаем элементы каждой строки на число, равное минимальному элементу данной строки. То же самое делаем и для столбцов. В получившейся матрице каждая строка и каждый столбец должны содержать хотя бы по одному нулю.

Этап 2. Ищем решение с нулевым значением. Для этого берем 1-ю строку; отметим один из нулей, остальные нули зачеркиваем и зачеркиваем нули в том столбце, где отметили ноль. То же делаем со всеми последующими строками.

Этап 3. Находим максимальное паросочетание следующим образом. Отмечаем всякую строку и всякий столбец с отмеченными нулями. В каждом из непомеченных столбцов найдем зачеркнутый ноль. В этой строке переходим к отмеченному нулю; в столбце отмеченного нуля ищем неотмеченный ноль в неотмеченной строке (если не нашли, проводим подобную рекурсивную процедуру для каждого зачеркнутого нуля в рассматриваемом столбце). Если нашли, то увеличиваем число паросочетаний: зачеркнутые нули отмечаем, отмеченные – зачеркиваем.

Если максимальное паросочетание дает насыщенную матрицу назначений, то решение найдено: номер строки каждого из отмеченных нулей соответствуют номеру претендента, а номер строки – номеру должности, на которое следует его назначить.

В противном случае переходим к Этапу 4.

Этап 4. Нахождение минимальной опоры – минимального множества рядов, содержащего все ее нули.

Помечаем всякую строку, которая не содержит отмеченных нулей (метки в Этапах 4, 5 и в Этапе 3 – разные). Помечаем всякий столбец, содержащий зачеркнутый ноль в каждой из отмеченных строк. Помечаем всякую строку, содержащую отмеченный ноль в каждом из помеченных столбцов. Повторяем эти две операции, пока процедура не исчерпает себя.

Строки берем помеченные, а столбцы – непомеченные, и выбранное отмечаем пунктиром. «Пунктирные» ряды и составляют минимальную опору.

Этап 5. Перестановка нулей.

Рассмотрим подматрицу, образованную элементами, не попавшими в опору. Возьмем минимальный элемент этой подматрицы. Вычтем это число из всех неотмеченных столбцов и прибавим ко всем неотмеченным строкам.

Этап 6. Переходим к Этапу 2.

Рассмотрим применение метода на примере. Пусть имеется 5 автомобилей такси и 5 заказов (рис. 1).

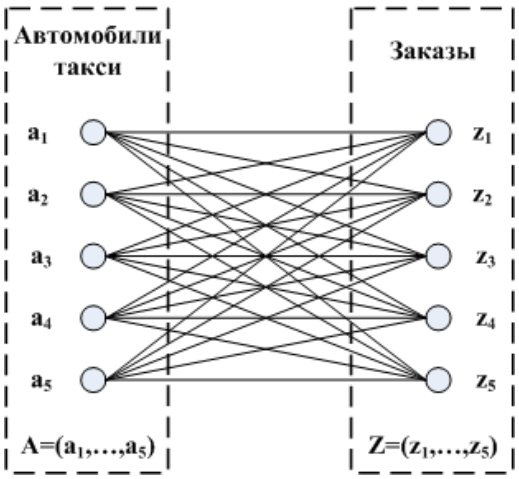

Рис. 1. *Граф соответствия «Автомобили – Заказы»*

Составим исходную таблицу (матрицу), табл. 1.

Таблица 1. *Исходная таблица соответствия*

Автомобиль	Заказ 1	Заказ 2	Заказ 3	Заказ 4	Заказ 5
1	2	4	1	3	3
2	1	5	4	1	2
3	3	5	2	2	4
4	1	4	3	1	4
5	3	2	5	3	5

Строкам соответствуют автомобили такси, а колонкам – заказы. Элементы матрицы показывают отдаленность автомобиля от места заказа в км.

После работы алгоритма, описанного выше, получим оптимальную матрицу назначений (табл. 2).

Таблица 2. *Оптимальная матрица назначений*

Автомобиль	Заказ 1	Заказ 2	Заказ 3	Заказ 4	Заказ 5
1			1		
2					1
3				1	
4	1				
5		1			

Таким образом получаем оптимальное распределение «Автомобиль – Заказ», рис. 2.

Минимальное значение целевой функции: 1+2+2+1+2=8, т. е. минимальный суммарный пробег автомобилей до мест заказов составит 8 км.

Разработка адаптивного алгоритма распределения заказов

Эффективным способом решения задачи о назначениях является венгерский метод [4–7] при котором осуществляется поиск максимального паросочетания в двудольном взвешенном графе, однако в общем случае $n \neq m$, и для поиска максимального паросочетания придется прямоугольную матрицу смежности размерности $n*m$ приводить к квадратной для выравнивания размерности, путем добавления пустых строк/столбцов, так чтобы $n=m$.

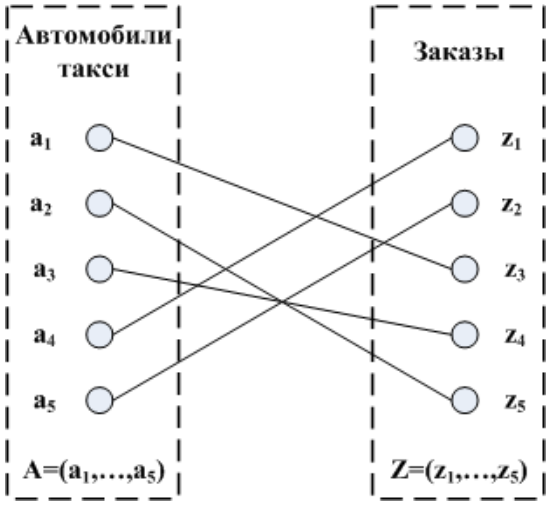

Рис. 2. *Оптимальное распределение «Автомобиль – Заказ»*

При этом может учитываться либо единственный критерий, например, суммарное время выполнения всех заказов, минимизация суммарного пробега автомобилей, максимальная выручка от выполнения заказов или др. В технологии распределения заказов могут учитываться особенности отдельных клиентов (приоритетный клиент, корпоративный, постоянный, проблемный, криминальный), а так же требования к приоритетности выполнения отдельных заказов (поездка в больницу, аэропорт, вокзал и др.).

Учитывая специфику работы таксопарка, целесообразно выравнивание размерности массивов A и Z производить путем предварительного уменьшения количества автомобилей или количества заказов по критерию $\min(m,n)$. В этом случае существенно уменьшается вычислительная сложность решаемой задачи.

Адаптивный алгоритм распределения заказов между автомобилями такси приведен на рис. 3.

Этап 1. Формирование двудольного графа $G=(Z,A)$, где $Z=\{z_1,\ldots,z_n\}$ – множество поступающих заказов и $A=\{a_1,\ldots,a_m\}$ – множество автомобилей такси, готовых их выполнить.

Этап 2. Определение стратегии распределения заказов в соответствии с оперативной ситуацией: $S=f(s_1,s_2,s_i,s_k)$, где факторы s_i определяют текущую загрузку, соотношение между количеством автомобилей и количеством заказов, общую размерность графа G и др. Если выбрана Стратегия 1, то выполняется Этап 3, в противном случае Этап 6.

Этап 3. Расчет локальных критериев эффективности, соответствующих ребрам графа G.

Этап 4. Преобразование графа G в граф G'. Целью преобразования является уменьшение мощности множества вершин Z или вершин множества A по критерию $\min(m,n)$.

Если в исходном графе G количество заказов превышает количество автомобилей, готовых их выполнить, т. е. $n>m$, то необходимо удалить из рассмотрения $(m-n)$ вершин множества Z.

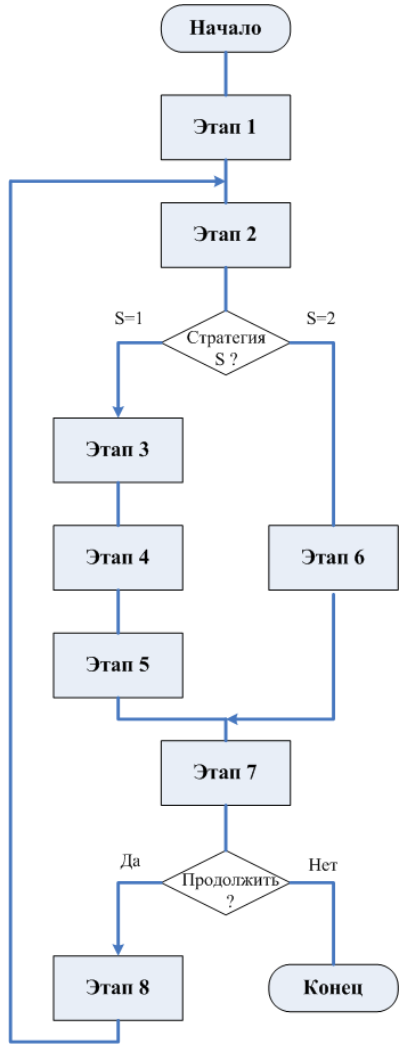

Рис. 3. *Адаптивный алгоритм распределения заказов такси*

Если в исходном графе G количество заказов меньше количества автомобилей, готовых их выполнить, т. е. $n<m$, то необходимо удалить из рассмотрения $n-m$ вершин множества A.

Этап 5. Решение задачи оптимального распределения заказов (множество вершин Z) между водителями такси (множество вершин A) венгерским методом.

Этап 6. Решение задачи максимального распределения заказов (множество вершин Z) между водителями такси (множество вершин A) алгоритмом Куна.

Этап 7. Формирование двудольного графа $G''=(Z'',A'')$ по критерию отказа водителей от выполнения заказа, где $A''\in A$ – множество водителей, отказавшихся от выполнения заказа, а $Z''\in Z$ – заказы, на выполнение которых пришел отказ от водителей.

Этап 8. Формирование нового графа G с учетом ограничений, накладываемых графом (Этап 1), или прекращение работы.

В результате выполнения предложенного алгоритма в каждый дискретный момент времени t_i формируется искомое множество паросочетаний P. Множества A' и Z', а также множества A'' (нераспределенные автомобили) и Z'' (нераспределенные заказы) переходят для рассмотрения в момент времени t_{i+1}.

На основе адаптивного алгоритма распределения заказов на обслуживание автомобилями такси разработано программное обеспечение на языке C#, используемое в работе информационно-телекоммуникационного комплекса «АГАТ». Это позволило сократить время на распределение заказов между автомобилями такси таксопарка от 10 до 25 %.

Выводы

На основе алгоритма Куна и венгерского метода нахождения максимального паросочетания разработан адаптивный алгоритм распределения заказов, включающий стратегию формирования базовой матрицы смежности вершин двудольного графа.

СПИСОК ЛИТЕРАТУРЫ

1. Перегудов Ф.И., Тарасенко Ф.П. Основы системного анализа. – Томск: Изд-во НТЛ, 1997. – 396 с.
2. Белов В.В., Сонькин Д.М. Некоторые задачи автоматизации городского такси с использованием микропроцессорных терминалов // Молодежь и современные информационные технологии: Сб. трудов VI Всеросс. научно-практ. конф. студентов, аспирантов и молодых ученых. – Томск, 26-28 февраля 2008. – Томск: СПб Графикс, 2008. – С. 18–19.
3. Колесников Л.А. Основы теории системного подхода. – Киев: Наукова думка, 1988. – 174 с.
4. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.
5. Свами М., Тхуласираман К. Графы, сети и алгоритмы. – М.: Мир, 1984. – 454 с.
6. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
7. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1985. – 512 с.

Поступила 05.11.2009 г.