УДК 004.021

А.Л. Гольдштейн

Пермский национальный исследовательский политехнический университет, Пермь, Россия

ЗАДАЧА О ВЫБОРЕ ЗАЯВОК

Рассматривается известная задача о выборе заявок или процессов, имеющая большое практическое значение. На основе сформулированных очевидных утверждений строится новый алгоритм решения данной задачи. Его идея состоит в работе со списками конфликтующих заявок. Каждой заявке приписан такой список, который на первой стадии алгоритма наполняется, а на второй сокращается до нуля. Обнуление всех списков свидетельствует о получении решения с максимальным числом не конфликтующих между собой заявок. Приводится оценка времени работы алгоритма, показывающая, что по этому показателю он не уступает известным сегодня алгоритмам. В то же время он отличается от них тем, что не требует предварительной сортировки заявок и в случае неединственности оптимального решения дает более плотное расписание (с меньшей суммой простоев).

Ключевые слова: задача выбора процессов, алгоритм выбора, расписание процессов.

A.L. Goldshtein

Perm National Research Polytechnic University, Perm, Russian Federation

ACTIVITY-SELECTION PROBLEM

Tis hpaper is related to a certain problem of activity-selection which is of great practical value. A new solution algorithm of the given problem is constructed on the basis of the obvious statements formulated. Its concept is to deal with the list of conflicting claims. The list which is filled at the first algorithm stage and reduced to zero at the second one is assigned to every claim. Nullification of all the lists indicates that the solution with the maximum number of claims which don't conflict with each other has been received. Algorithm operation time has been estimated proving that according to this index it is not inferior to currently known algorithms. At the same time it is different from them because it does not require preliminary claim sorting and in case an optimal solution is not unique it will give denser schedule (with less sum of idle time).

Keywords: activity-selection problem, selection algorithm, process schedule.

Важнейшей особенностью современных задач принятия решений является стремление к поиску наилучших решений. В свою очередь задачи оптимизации могут решаться разными методами и алгоритмами, отличающимися эффективностью и точностью. Возможность выбора позволяет исследователю подобрать наиболее подходящий для конкретной задачи алгоритм.

В настоящей статье предлагается новый алгоритм для известной задачи, которую называют задачей о выборе процессов или задачей о выборе заявок. Эта задача формулируется следующим образом. Имеется конечное множество процессов, операций, работ или заявок $Z = \{z_1, z_2,\ldots,z_n\}$, претендующих на один и тот же неделимый ресурс. Для каждого процесса z_i известны начальный момент s_i и конечный момент f_i, такие что $0 \le s_i < f_i < \infty$. Если временные интервалы двух или более процессов пересекаются, то они являются конкурирующими или несовместимыми. Задача заключается в выборе такого подмножества $Z^* \subseteq Z$ взаимно совместимых процессов, которое имеет максимальный размер.

Очевидно, что элементами множества Z могут быть не только собственно процессы в компьютерах. Приведем несколько примеров, соответствующих рассматриваемой задаче.

1. В некотором городе проходит фестиваль кинофильмов или театральных постановок, который проводится одновременно в разных залах. Имея расписание всех мероприятий, некий журналист или критик стремится составить для себя такой непротиворечивый график, который позволит ему посмотреть как можно больше фильмов или спектаклей.

2. При составлении расписания учебных занятий в вузе часто возникает ситуация, когда на одну аудиторию претендуют n несовместимых по времени занятий, начало и окончание которых уже фиксированы. Диспетчер пытается найти вариант расписания, по которому в аудитории пройдет максимальное число занятий.

3. В пункте проката имеется дефицитный инструмент (спортинвентарь, станок и т.п.). Поступило n заявок на взятие этого инструмента в прокат с желаемыми сроками использования. Сходная проблема имеет место при сдаче квартиры (комнаты, дома) в популярной курортной зоне в летние месяцы, когда поступает много не совместимых по времени заявок. В таких случаях понятно желание хозяина пункта проката (квартиры) удовлетворить максимальное количество заявок.

Можно привести и другие примеры проблемных ситуаций, приводящие к задаче выбора заявок.

Для решения задачи выбора процессов разработан ряд эффективных алгоритмов, в том числе несколько вариантов жадных алгоритмов [1, 2]. Применение этих алгоритмов предполагает, что исходное множество заявок уже упорядочено по f_i или s_i. Ниже приводится еще один алгоритм, не требующий предварительной сортировки.

Построение алгоритма

Предлагаемый алгоритм строится на основе следующих утверждений.

Утверждение 1. Заявка $z_i \in Z$, совместимая со всеми заявками из $Z\backslash z_i$, принадлежит оптимальному списку $Z^* \subseteq Z$.

Утверждение 2. Заявка $z_i \in Z$, имеющая наибольшее число несовместимых с ней заявок из $Z\backslash z_i$, не может принадлежать $Z^* \subseteq Z$.

Заявкам в исходном множестве Z присвоены номера в естественном порядке, то есть какое-либо упорядочение Z не производится. Максимальный номер заявки равен числу заявок n в исходном множестве Z, то есть $n = |Z|$. В процессе работы алгоритма при сокращении текущего Z соответственно уменьшается и n. Каждой заявке припишем список (множество) M_i, содержащий номера заявок, не совместимых с z_i. Первоначально все списки пустые. На первом этапе работы алгоритма эти списки последовательно пополняются, а на втором происходит их сокращение в соответствии с утверждением 2.

Алгоритм:

1.Выполняется проверка на совместимость заявок и заполнение списков M_i.

1.1. $n=|Z|$, $i=1$

1.2. $j= i+1$

1.3. если $f_i \le s_j$, то переход на 1.5,

иначе если $f_j \le s_i$, то переход на 1.5

1.4. $M_i= M_i \cup \{j\}$

$M_j= M_j \cup \{i\}$

1.5. если $j < n$, то $j= j+1$ и переход на 1.3,

иначе если $i < n–1$, то $i= i+1$и переход на 1.2

2. Сокращается множество заявок до получения искомого решения.

2.1. $m=\max| M_i |, \ i = \overline{1,n}$

2.2. если $m>0$, то $N=\{$номера заявок с $| M_i|=m\}$,

иначе $Z^* = Z$, конец

2.3. $k=\arg\max(f_i – s_i), \ i \in N$

2.4. $Z=Z\backslash z_k$,

для $\forall i$ такого, что $k \in M_i$, $M_i= M_i\backslash k$

2.5. $n= n–1$, переход на 2.1.

Выполнение одного из неравенств в п.1.3 означает, что сравниваемые заявки z_i и z_j совместимы, а в противном случае номер j зано-

сится в список M_i, а номер i в M_j. По мере выполнения действий 1.2–1.5 списки M пополняются строго возрастающими номерами заявок. Во второй части алгоритма из Z удаляется заявка с максимальным числом несовместимостей (вместе со своим списком M_k). Если таких заявок больше одной, то есть $|N|>1$, то удаляется заявка с максимальной продолжительностью (п. 2.3).

Оценим время работы алгоритма как функцию мощности исходного множества Z. Решающее влияние n на время работы алгоритма проявляется в его первой части. Здесь алгоритм выполняет парные сравнения заявок, а их количество непосредственно зависит от n. Так как число сочетаний $C_n^2 = \frac{n(n-1)}{2}$, то оценкой времени работы приведенного выше алгоритма справедливо считать $O(n^2)$. Время работы известных алгоритмов решения задачи о заявках оценивается как $O(n)$, но оно относится к случаю упорядоченного по одному из моментов (s_i или f_i) множества Z [1]. Если же учесть, что сортировка слиянием оценивается временем $O(n \lg n)$, а по методу вставки – $O(n^2)$, становится очевидным, что предложенный алгоритм не уступает приведенным в [1, 2].

Тестирование алгоритма

В программной реализации алгоритма множество заявок формируется с помощью датчика случайных чисел с интервалом [0, 100]. Для образования заявки генерируются 2 числа, и большее значение присваивается f, а меньшее s.

Исследование работы алгоритма проведено для различных значений n, от 30 до 300. Прежде всего, нас интересовал характер зависимости времени работы алгоритма от n. С этой целью проведено 10 серий прогона алгоритма, в каждой серии генерировалось 15 множеств Z и замерялось время решения соответствующих задач. В качестве показателя работы алгоритма взята оценка среднего времени решения задачи $T_{ср}$. Полученные результаты приведены в табл. 1.

Чтобы исключить влияние характеристик компьютера, $T_{ср}$ дано в относительных единицах (за единицу принято среднее время при n = 30). Там же в относительных единицах представлены оценки среднеквадратичного отклонения σ_T.

Таблица 1

Результаты тестирования алгоритма

n	30	60	90	120	150	180	210	240	270	300
$(n/30)^2$	1	4	9	16	25	36	49	64	81	100
$T_{\text{ср}}$	1	3,324	7,117	14,716	23,457	32,617	46,120	60,289	79,093	95,775
σ_T	0,166	0,509	0,666	2,780	1,983	2,871	2,297	2,657	4,373	3,985

Из этой таблицы следует, что время работы алгоритма с ростом n увеличивается не быстрее чем $(n/30)^2$. Время работы алгоритма определялось и при увеличении интервала датчика случайных чисел в два раза в том же диапазоне изменения n, при этом характер зависимости от n не претерпел существенных изменений.

Также сравнивались решения, получаемые по нашему алгоритму (NA) и алгоритму Greedy_Activity_Selector (GAS) [1]. В процессе исследования была установлена существенная особенность алгоритма NA. В тех случаях, когда задача имеет более одного решения, расписание по предложенному алгоритму получается более плотным, чем по GAS, то есть в нем меньше простоев между соседними заявками. Для подтверждения этого свойства алгоритма NA приведем два примера. В табл. 2 представлено множество Z из 30 заявок, параметры которых выдал датчик случайных чисел.

Таблица 2

Исходное множество заявок (пример 1)

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_i	32	25	99	52	85	30	38	72	92	18	76	84	70	29	7
f_i	52	100	100	93	93	74	44	92	95	34	78	91	84	39	46

Окончание табл. 2

i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
s_i	12	71	42	32	39	22	58	61	42	53	25	36	70	49	33
f_i	41	95	77	98	53	84	65	74	43	64	64	70	99	80	91

Решения для Z из примера 1, полученные по обоим алгоритмам, приведены в табл. 3. Сравнение показывает, что при одинаковом числе заявок в обоих расписаниях сумма простоев по алгоритму GAS составляет 41 единицу времени, а по алгоритму NA только 16.

Таблица 3

Результаты работы алгоритмов для примера 1

Алгоритм							
GAS				NA			
i	s_i	f_i	Простои	i	s_i	f_i	Простои
10	18	34		10	18	34	
24	42	43	8	20	39	53	5
25	53	64	10	25	53	64	0
11	76	78	12	13	70	84	6
12	84	91	6	12	84	91	0
9	92	95	1	9	92	95	1
3	99	100	4	3	99	100	4

Данные для второго примера, сгенерированные также датчиком случайных чисел, представлены в табл. 4.

Таблица 4

Исходное множество заявок (пример 2)

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
s_i	50	53	51	14	81	48	2	43	25	95	99	91	37	77	54	61	89	74	35	74
f_i	66	78	61	32	93	95	57	80	55	98	100	93	62	83	66	79	93	99	48	87

Окончание табл. 4

i	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
s_i	97	90	65	35	49	22	34	71	45	97	16	9	37	60	85	25	83	17	47	37
f_i	98	97	91	49	56	98	56	74	94	99	68	36	52	100	92	87	90	48	64	67

Применив сравниваемые алгоритмы к множеству заявок из табл. 4, получили расписания, представленные в табл. 5.

Таблица 5

Результаты работы алгоритмов для примера 2

Алгоритм							
GAS				NA			
i	s_i	f_i	Простои	i	s_i	f_i	Простои
4	14	32		4	14	32	
19	35	48	3	24	35	49	3
15	54	66	6	1	50	66	1
28	71	74	5	28	71	74	5
14	77	83	3	14	77	83	3
37	83	90	0	37	83	90	0
12	91	93	1	22	90	97	0
10	95	98	2	30	97	99	0
11	99	100	1	11	99	100	0

Как следует из табл. 5, суммарный простой по расписанию GAS равен 21 единице времени, а по расписанию NA только 12, то есть почти в два раза меньше.

Эти примеры показывают, что алгоритм NA одновременно с нахождением максимального числа совместимых заявок обеспечивает более эффективное использование времени на рассматриваемом интервале, чем применяемый сегодня алгоритм GAS. Эта особенность алгоритма может иметь решающее значение в некоторых приложениях. Так, при сдаче в аренду помещения или оборудования, если цена аренды за единицу времени постоянна, выбор заявок по алгоритму NA при охвате максимального числа клиентов даст больший доход владельцу помещения (оборудования).

Задача о выборе заявок или процессов относится к классу комбинаторных задач. Для ее решения сегодня применяются жадные алгоритмы, которые могут работать только после предварительной сортировки заявок по начальным или конечным моментам [1, 2]. Эти алгоритмы работают по принципу «снизу вверх», последовательно наращивая результирующее множество до получения оптимального решения. Предложенный в настоящей работе алгоритм NA не требует сортировки заявок и работает по принципу «сверху вниз», последовательно сужая исходное множество заявок до достижения искомого. По времени решения задачи сравниваемые алгоритмы имеют близкие оценки, но алгоритм NA будет предпочтительнее, когда сортировка исходного множества заявок (процессов) нежелательна или невозможна. А в тех случаях, когда требуется не только набрать максимальное число совместимых заявок, но и при этом получить наибольшую плотность расписания, построенный алгоритм NA, как показывают результаты сравнения, оказывается явно предпочтительнее известных жадных алгоритмов.

Библиографический список

1. Алгоритмы: построение и анализ: пер. с англ. / Т.Х. Кормен, Ч.И. Лейзерсон, Р.Л. Ривест, К. Штайн. – 2-е изд. – М.: Вильямс, 2009.

2. Lawler E.L. Combinatorial Optimization: Networks and Matroids. – Holt, Rinehart and Winston, 1976.

Сведения об авторе

Гольдштейн Аркадий Леонидович (Пермь, Россия) – кандидат технических наук, профессор кафедры информационных технологий и автоматизированных систем Пермского национального исследовательского политехнического университета (614990, Пермь, Комсомольский пр., 29, e-mail: gal@pstu.ru).

About the author

Goldshtein Arkady Leonidovich (Perm, Russian Federation), PhD of technical Science, Professor at the department of information technology and automated systems of Perm National Research Polytechnic University (614990, Perm, 29, Komsomolsky pr., e-mail: gal@pstu.ru).

Получено 06.09.2013