© 2013 г. Д.И. КОГАН, д-р техн. наук
(Московский государственный университет приборостроения и информатики, Москва),
А.С. КУИМОВА,
Ю.С. ФЕДОСЕНКО, д-р техн. наук
(Волжская государственная академия водного транспорта, Нижний Новгород)

ЗАДАЧИ ОБСЛУЖИВАНИЯ БИНАРНОГО ПОТОКА ОБЪЕКТОВ В СИСТЕМЕ С НАКОПИТЕЛЬНО-РАСХОДНЫМ КОМПОНЕНТОМ[1]

Рассматривается модель одностадийного обслуживания конечного детерминированного потока объектов процессором с накопительно-расходным компонентом — резервуаром ограниченной емкости. Поток состоит из подпотока объектов, пополняющих резервуар, и подпотока объектов, заполняемых из резервуара. С каждым объектом ассоциируется линейная функция индивидуального штрафа за время пребывания в системе обслуживания. Изучается задача построения расписания, минимизирующего суммарный штраф по всем объектам потока. Конструируемые алгоритмы основываются на принципе динамического программирования, схеме ветвей и границ, а также их совместной реализации. Приводятся результаты вычислительных экспериментов.

1. Введение

Рассматриваемая модель возникла при изучении процессов грузовой обработки танкерного флота в условиях Северного завоза [1] через речной порт г. Салехарда.

В течение непродолжительного навигационного периода крупнотоннажным танкерным флотом по реке Обь с нефтеперегонных заводов Западной Сибири в Салехардский порт доставляется дизельное топливо. Прибывающие суда в определенной очередности подаются к специализированному терминалу, техническими средствами которого нефтепродукт перекачивается в резервуар ограниченной ёмкости. Многочисленные пункты потребления дизельного топлива, являющегося основным энергетическим ресурсом в Заполярье, располагаются, в основном, по берегам Обской губы и малых рек прилегающего региона полуострова Ямал. Доставка топлива в эти пункты осуществляется водным путем из Салехардского порта малотоннажными танкерами ледового класса, загрузка которых осуществляется на вышеупомянутом специализированном терминале; по техническим условиям на этом терминале не может одновременно обслуживаться более одного танкера.

[1]Работа выполнена при финансовой поддержке Фонда научно-исследовательской деятельности Волжской государственной академии водного транспорта (грант № 02-2013).

В описанной схеме Северного завоза задействованы танкеры, характеризующиеся различными технико-экономическими параметрами. Задача диспетчеризации заключается в выработке стратегии управления очередностью их грузовой обработки — расписания обслуживания, которое в пределах горизонта оперативного планирования обеспечивает сокращение суммарных эксплуатационных расходов, обусловленных непроизводительными простоями флота.

Специфика задач диспетчеризации заключается в том, что между моментом, когда полностью определились исходные данные конкретной задачи, и моментом, когда следует начать работу по построенному расписанию, проходит относительно небольшой промежуток времени. В течение этого промежутка задача должна быть решена. Поэтому при математическом исследовании каждой типовой задачи диспетчеризации важно получить оценку ее вычислительной сложности. Вячеслав Сергеевич Танаев был одним из первых математиков, активно исследовавших вопросы вычислительной сложности задач теории расписаний; в написанных им совместно с коллегами и получивших мировое признание монографиях [2–4] вопросы разделения задач на полиномиально разрешимые и NP-трудные [5] играют ключевую роль.

Статья состоит из шести разделов и Приложения. В следующем за Введением разделе 2 приводится описание модели обслуживания, дана математическая постановка задачи синтеза оптимального расписания обслуживания, сформулированы связанные с этой задачей результаты о труднорешаемости. Раздел 3 посвящен описанию процедуры решения поставленной задачи на основе рекуррентных соотношений динамического программирования; технология решения этой задачи с использованием схемы ветвей и границ приводится в разделе 4. В разделе 5 рассмотрены некоторые пути сокращения счета при синтезе расписаний обслуживания, в том числе благодаря комбинированному применению концепции динамического программирования и схемы ветвей и границ. В этом же разделе вводится конкретизация рассматриваемой задачи, предполагающая ограниченность числа типов подлежащих обслуживанию объектов; получаемая при таком ограничении частная задача оказывается полиномиально разрешимой. Раздел 6 — Заключение. Доказательства приведенных в разделе 2 теорем о труднорешаемости вынесены в Приложение.

2. Математическая модель обслуживания, постановка оптимизационной задачи и исследование вычислительной сложности

Изучается модель $\boldsymbol{M}$, в которой конечный поток объектов $O_n = \{o_1, o_2, \ldots, o_n\}$ подлежит однофазному обслуживанию стационарным процессором с накопительно-расходным компонентом — резервуаром ёмкости V^*. Для каждого объекта o_i, $i = \overline{1,n}$ определены целочисленные параметры: t_i — момент поступления в очередь на обслуживание; τ_i — норма длительности обслуживания; a_i — штраф за единицу времени пребывания в системе обслуживания; v_i — объемная характеристика (вместимость объекта). Поток O_n состоит из подпотоков O^+ и O^-. Объекты подпотока O^+ предназначены для пополнения резервуара, объекты подпотока O^- — для заполнения из резервуара. Принадлежность объекта o_i, $i = \overline{1,n}$ тому или иному подпотоку определяется значением параметра w_i ($w_i = +1$, если $o_i \in O^+$; $w_i = -1$, если $o_i \in O^-$). Объекты пронумерованы в порядке их поступления: $0 = t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$.

Заполнение резервуара в момент времени t будем характеризовать переменной $V(t)$ с известным начальным значением $V(0)$. Обслуживание объекта o_i из подпотока O^+ может быть начато при наличии в резервуаре достаточного свободного объема; в результате реализации обслуживания такого объекта заполнение резервуара увеличивается на величину v_i. Объект o_i из подпотока O^- может быть начат обслуживанием при наличии достаточного заполнения резервуара; в результате реализации его обслуживания заполнение резервуара уменьшается на величину v_i. Считается, что процессор не может обслуживать более одного объекта одновременно; обслуживание каждого объекта осуществляется без прерываний.

Расписание обслуживания потока O_n определяем как перестановку $\rho = \{i(1), i(2), \ldots, i(n)\}$ совокупности индексов $N = \{1, 2, \ldots, n\}$; при его реализации объект с индексом $i(k)$ обслуживается k-м по очереди ($k = \overline{1, n}$). Расписание ρ именуем допустимым, если в процессе его выполнения удовлетворяются отмеченные выше объемные ограничения, т.е. $0 \leqslant V(0) + \sum_{p=1}^{q} v_{i(p)} \cdot w_{i(p)} \leqslant V^*$, $q = 1, 2, \ldots, n$. Совокупность допустимых в модели $\boldsymbol{M}$ расписаний обслуживания обозначим Ω.

Как очевидно, заполнение резервуара после завершения обслуживания всех объектов потока O_n оказывается равным $V(0) + \sum_{i=1}^{n} v_i \cdot w_i$. При этом выполнение неравенств

$$0 \leqslant V(0) + \sum_{i=1}^{n} v_i \cdot w_i \leqslant V^* \tag{1}$$

является необходимым, но недостаточным условием непустоты множества Ω. Приведем иллюстрирующий пример: положим $V(0) = V^* = 5$; подпоток O^+ составляет единственный объект с объемной характеристикой, равной 5; в подпоток O^- входит пять объектов, объемная характеристика каждого из них равна 2. Здесь условие (1) выполнено, но допустимого расписания обслуживания не имеется.

Выделим и назовем моделью $\boldsymbol{M}_0$ частый случай модели $\boldsymbol{M}$, в котором все подлежащие обслуживанию объекты изначально присутствуют в системе: $t_k = 0$, $k = 1, 2, \ldots, n$.

Т е о р е м а 1. *Проблема определения по исходным данным модели обслуживания $\boldsymbol{M}_0$ (условие (1) выполнено), является ли множество допустимых в ней расписаний обслуживания непустым, NP-полна в сильном смысле.*

Доказательство теоремы 1 приведено в Приложении.

Легко видеть, что при выполнении неравенств (1) достаточным условием непустоты совокупности Ω является выполнение следующей системы неравенств

$$2 \cdot v_i \leqslant V^*, i = 1, 2, \ldots, n. \tag{2}$$

В рассматриваемом классе воднотранспортных приложений неравенства (1) и (2) всегда имеют место. Далее будем считать их выполненными. Заметим также, что условия

$$\{\sum_{i: o_i \in O^-} v_i \leqslant V(0)\} \,\&\{\sum_{i: o_i \in O^+} v_i \leqslant V^* - V(0)\} \tag{3}$$

необходимы и достаточны для того, чтобы любое расписание обслуживания было допустимо.

Считаем, что процессор готов к обслуживанию объектов начиная с момента времени $t = 0$. Каждое допустимое расписание $\rho = \{i(1), i(2), \ldots, i(n)\}$ однозначно определяет для произвольного объекта $o_{i(k)}$ моменты начала и завершения его обслуживания, которые далее будут обозначаться $t_{beg}(i(k), \rho)$ и $t^*(i(k), \rho)$ соответственно. Указанные моменты последовательно, в порядке возрастания значений параметра k вычисляются по формулам

$$t_{beg}(i(1), \rho) = t_{i(1)};$$
$$t_{beg}(i(k), \rho) = \max[t^*(i(k-1), \rho), t_{i(k)}], k = 2, 3, \ldots, n;$$
$$t^*(i(k), \rho) = t_{beg}(i(k), \rho) + \tau_{i(k)}, k = 1, 2, \ldots, n.$$

При реализации расписания ρ суммарный штраф $K(\rho)$ по всем объектам потока O_n оказывается равным $\sum_{i=1}^{n} a_i \cdot [t^*(i, \rho) - t_i]$.

Задача 1. *Найти допустимое расписания обслуживания, минимизирующее величину суммарного штрафа*

$$\min_{\rho \in \Omega} K(\rho). \tag{4}$$

Задачей 1_0 назовем задачу 1, сформулированную в рамках частной модели $\boldsymbol{M}_0$, т.е. для случая, когда все подлежащие обслуживанию объекты изначально присутствуют в системе: $t_k = 0$, $k = 1, 2, \ldots, n$.

Теорема 2. *Задача 1_0 NP-трудна в сильном смысле.*

Доказательство теоремы 2 приведено в Приложении.

3. Решение задачи 1 методом динамического программирования

Оптимальное значение критерия задачи 1 обозначим K_{opt}. Пусть $R(t)$ — определяемое исходными данными задачи 1 подмножество индексов объектов потока O_n, которые поступают в систему обслуживания в момент дискретного времени t; при этом $R(0)$ — подмножество индексов объектов, ожидающих обслуживания по состоянию на момент времени $t = 0$. Совокупность индексов объектов, прибывающих в систему на отрезке времени $[t+1, t+\Delta]$, где $\Delta \geqslant 1$, обозначим через $D(t, \Delta)$. Таким образом, $D(t, \Delta) = \bigcup_{i=i}^{\Delta} R(t+i)$.

Состояние рассматриваемой системы в каждый очередной момент t принятия решения по загрузке процессора работой (определяется, какой объект из числа ожидающих принять на немедленное обслуживание) однозначно характеризуется парой (t, Q), где Q — совокупность индексов объектов, которые к моменту времени t поступили, но пока не обслужены; в момент времени t процессор свободен. Следует отметить, что по известному множеству Q и значению t заполнение резервуара $V(t, Q)$ определяется однозначно. Через $Z(t, Q)$ обозначим частную задачу, получаемую из исходной в предположении, что к моменту времени t необходимо завершить обслуживанием все поступившие объекты за исключением тех, чьи индексы входят в совокупность Q; в качестве минимизируемого критерия выступает суммарный штраф по объектам, завершенным обслуживанием не позднее момента времени t. Оптимальное значение критерия в задаче $Z(t, Q)$ обозначим $K(t, Q)$.

В множество необслуженных объектов Q всегда будем дополнительно включать фиктивный (нулевой) объект o_0 с характеристиками $a_0 = 0$, $\tau_0 = 1$, $v_0 = 0$; для определенности полагаем $w_0 = +1$. Обслуживание фиктивного объекта означает простой процессора в течение одного такта. Простои процессора объективно имеют место, если: а) все ранее поступившие объекты уже обслужены, а другие объекты потока пока не поступили; б) ни один из ранее поступивших и ожидающих обслуживания объектов на обслуживание с момента времени t принят быть не может по причине отсутствия в резервуаре достаточного свободного объема для его пополнения (для объектов подпотока O^+) или достаточного объёма для заполнения из резервуара (для объектов подпотока O^-); в) целесообразно не занимать процессор обслуживанием ни одним из ранее поступивших и ожидающих обслуживания объектов с тем, чтобы обеспечить приоритетное обслуживание некоторого, пока не поступившего объекта с высоким значением штрафа за единицу времени простоя. Факт наличия фиктивного объекта в множестве Q при перечислении элементов этого множества ниже специально отмечаться не будет.

В текущем состоянии (t, Q) объект o_i, $i \in Q$ допустим к обслуживанию только при выполнении одного из условий: $(w_i = +1)\,\&(V(t,Q) + v_i \leqslant V^*)$ или $(w_i = -1)\,\&(V(t,Q) \geqslant v_i)$. Множество индексов объектов, допустимых к обслуживанию в состоянии (t, Q), обозначим через $Q^*(t, Q)$. Отметим, что обслуживание фиктивного объекта допустимо всегда: $0 \in Q^*(t, Q)$. После выполнения обслуживания объекта o_i, $i \in Q^*(t, Q)$ заполнение резервуара становится равным $V(t, Q) + w_i v_i$.

Состояние (t', Q') назовем непосредственно предшествующим состоянию (t, Q), если в множестве $Q^*(t', Q')$ имеется объект o_α такой, что $t = t' + \tau_\alpha$ и $Q = (Q' \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$. Если в состоянии (t', Q') в качестве очередного обслуживаемого взят объект o_α, то (t, Q) — следующее состояние, в котором будет приниматься решение по загрузке процессора. Через $\alpha(t', Q', t, Q)$ будем обозначать номер объекта, обслуживание которого влечет переход системы из состояния (t', Q') в состояние (t, Q). Множество всех непосредственно предшествующих (t, Q) состояний будем записывать $N(t, Q)$.

Состояние (t, Q) назовем непосредственно следующим за (t', Q') состоянием, если (t', Q') непосредственно предшествует состоянию (t, Q). Состояния, непосредственно следующие за состояниями вида $(t_n + \theta, \{i\})$, где θ — произвольная положительная константа, а $i \in \{1, 2, \ldots, n\}$, будем именовать финальными. Множество всех финальных состояний обозначим $\mathbf{F}$.

В принятых обозначениях рекуррентные соотношения динамического программирования для решения задачи 1 записываются в виде

$$K(0, R(0)) = 0, \tag{5}$$

$$K(t, Q) = \min_{(t', Q') \in N(t,Q)} [K(t', Q') + a_{\alpha(t',Q',t,Q)}(t - t_{\alpha(t',Q',t,Q)})], \tag{6}$$

$$K_{opt} = \min_{(t, \varnothing) \in \mathbf{F}} [K(t, \varnothing)]. \tag{7}$$

Приведенные соотношения ориентированы на реализацию схемы прямого счета (forward dynamic programming [6]), т.е. без рассмотрения состояний, недостижимых из начального.

Опишем обозначаемый далее символом $\mathbf{A}$ алгоритм вычислений по формулам (5)–(7).

Предварительно определим, что раскрытие состояния $S = (t', Q')$ — это операция порождения всех записей, определяющих непосредственно следующие за S состояния; каждая такая запись имеет вид $[t', Q', \alpha, t, Q]$, где $\alpha = \alpha(t', Q', t, Q)$.

Алгоритм **A** действует следующим образом (вводимые при описании алгоритма списки G_1–G_4 считаем изначально пустыми).

Этап 1: а) фиксируется $K(0, R(0)) = 0$; б) раскрывается начальное состояние $(0, R(0))$, получаемые при этом пятикомпонентные записи включаем в список G_1.

Этап 2: из списка G_1 изымаются и переносятся в список G_2 все записи с минимальным значением четвертой компоненты (параметра t); если список G_1 пуст, переходим к этапу 5.

Этап 3: для каждой пары (t, Q), присутствующей в записях списка G_2 в качестве четвертой-пятой компонент (отметим, что одна пара (t, Q) может входить в качестве указанных компонент в несколько записей): а) по формуле (6) вычисляется значение $K(t, Q)$; при этом определяется состояние (t^*, Q^*), на котором реализуется минимум правой части соотношения (6) и из которого следует выполнить непосредственный переход в состояние (t, Q) при реализации оптимального в задаче $Z(t, Q)$ расписания; определяется также соответствующий индекс α^*, $\alpha^* = (t^*, Q^*, t, Q)$; б) запись $[t^*, Q^*, \alpha^*, t, Q, K(t, Q)]$ вносится в список G_3; если пара (t, Q) является финальной, то тройка $\{t, \varnothing, K(t, \varnothing)\}$ дополнительно вносится в список G_4.

Этап 4: а) для каждого нефинального состояния — пары (t, Q), присутствующей в записях списка G_3 в качестве четвертой–пятой компонент, выполняется процедура раскрытия, получаемые при этом пятикомпонентные записи включаем в список G_1; б) опустошается список G_2 и осуществляется переход к этапу 2.

Этап 5: в построенном списке G_4 определяется запись $\{t^+, Q^+, K(t^+, Q^+)\}$ с наименьшим значением третьей компоненты; $K(t^+, Q^+)$ — оптимальное значение критерия в задаче 1, $K(t^+, Q^+) = K_{opt}$.

Отметим, что наличие списка G_3 позволяет элементарным образом определить для задачи 1 оптимальное расписание обслуживания объектов.

Изложенный алгоритм решения задачи 1 имеет экспоненциально зависящую от количества подлежащих обслуживанию объектов оценку вычислительной сложности — экспоненциально от n зависит число состояний системы. Данный факт соответствует приведенному выше результату об NP-трудности изучаемой задачи.

4. Решение задачи 1 методом ветвей и границ

Реализация метода ветвей и границ [7, 8] состоит в построении достаточного для определения оптимального решения фрагмента дерева вариантов. Вершины этого дерева соответствуют состояниям (t, Q) системы обслуживания; в частности, его корень соответствует состоянию $(0, R(0))$. Далее именами вершин будем считать названия соответствующих состояний.

Вычислительная процедура, реализующая схему ветвей и границ, полностью определяется заданием способов: а) ветвления, б) получения верхних оценок значений функции суммарного штрафа (UB) в конструируемых вершинах дерева вариантов, в) получения нижних оценок значений функции суммарного штрафа (LB) в конструируемых вершинах дерева вариантов, г) выбора вершины для очередного ветвления.

Ветвление в вершине — аналог раскрытия состояния при решении задачи методом динамического программирования. Напомним, что раскрытие состояния $S = (t', Q')$ — это порождение всех записей, определяющих непосредственно следующие за S состояния; каждая такая запись имеет вид $[t', Q', \alpha, t, Q]$, где α — индекс объекта, обслуживание которого переводит систему из состояния (t', Q') в состояние (t, Q). При ветвлении в вершине (t', Q') каждая порождаемая при раскрытии одноименного состояния запись $[t', Q', \alpha, t, Q]$ отображается носящей имя α дугой, проводимой из вершины (t', Q') во вновь вводимую вершину (t, Q).

При подсчете в произвольной вершине (t, Q) соответствующих ей верхней и нижней оценок используем тот факт, что последовательность $\rho(t, Q)$ обслуживания объектов o_i, по состоянию на момент времени t прибывших в систему и таких, что $i \notin Q$, уже определена. Эту последовательность задают имена дуг, образующих в построенном фрагменте дерева вариантов путь от его корня до вершины (t, Q).

При получении оценок будет использоваться алгоритм решения следующей простейшей задачи. Требуется найти минимизирующее суммарный штраф расписание обслуживания одним процессором множества объектов $1, 2, \ldots, n$ в ситуации, когда каждый объект o_i характеризуется двумя показателями: τ_i — продолжительность обслуживания; a_i — штраф за единицу времени пребывания в системе обслуживания; все объекты готовы к обслуживанию начиная от момента времени $t = 0$. Решающий алгоритм [2, 4]) для каждого объекта i вычисляет показатель $\mu_i = a_i/\tau_i$, назовем этот показатель μ-характеристикой объекта; далее объекты упорядочиваются по убыванию их μ-характеристик; полученная последовательность является оптимальным расписанием обслуживания. Изложенный алгоритм, назовем его μ-алгоритмом, в качестве эвристического адаптируется для обобщения, в котором подлежащие обслуживанию объекты поступают во времени, т.е для каждого объекта o_i дополнительно задается момент t_i готовности к обслуживанию. В каждый очередной момент принятия решения из совокупности ожидающих следует выбрать и принять на немедленное обслуживание объект с максимальной μ-характеристикой. Получаемое для обобщенной задачи расписание назовем μ-расписанием. Методом от противного показывается, что если при реализации μ-расписания в период обслуживания любого объекта i не поступает объект k такой, что $\mu_k > \mu_i$, то построенное μ-расписание оптимально.

Для получения в произвольной вершине (t, Q) верхней оценки UB достраиваем последовательность $\rho(t, Q)$ следующим алгоритмом: в состоянии (t, Q) из совокупности $Q^*(t, Q)$ выбираем индекс j, обеспечивающий максимально возможное значение μ-характеристики; определяем, что в момент времени t начинается обслуживание объекта o_j; идентичным образом определяем очередной обслуживаемый объект для каждого из последующих состояний системы. Пусть $\rho^*(t, Q)$ — полученное в итоге n-элементное расписание. Тогда $K(\rho^*(t, Q))$ — соответствующая вершине (t, Q) верхняя оценка UB.

При отыскании для произвольной вершины (t, Q) соответствующей ей нижней оценки LB действуем следующим образом.

Принимаем $\rho(t, Q)$ как начальную часть расписания обслуживания. Далее отказываемся учитывать ёмкостные характеристики объектов (становится неважным, какому подпотоку принадлежит тот или иной объект) и полагаем, что все объекты дробимы (расщепляемы). В состоянии (t, Q) из совокупности Q выбираем индекс j, определяющий объект с наибольшей μ-характеристикой. Обслуживание объекта o_j начинается от момента времени t. При этом возможны два случая: 1) в процессе обслуживания o_j в систему не поступит объект o_k с большим значением μ-характеристики; 2) такой объект в систему поступит. В первом случае полагаем, что обслуживание объекта o_j реализуется без прерываний и завершается в момент $t + \tau_j$. Во втором случае выполняем расщепление объекта o_j на два объекта, $o_{j[1]}$ и $o_{j[2]}$. При этом $\tau_{j[1]}$ и $\tau_{j[2]}$ полагаются равными $t_k - t$ и $\tau_j - (t_k - t)$ соответственно; $a_{j[1]}$ и $a_{j[2]}$ считаем равными $a_j \cdot \tau_{j[1]}/\tau_j$ и $a_j \cdot \tau_{j[2]}/\tau_j$ соответственно. Обслуживание объекта $o_{j[1]}$ завершается в момент времени t_k; объект $o_{j[2]}$ включается в совокупность объектов, ожидающих обслуживания по состоянию на момент принятия решения t_k.

Действуя идентичным образом, достраиваем расписание от состояния, получаемого на момент $t+\tau_j$ при реализации первого случая, и от состояния, получаемого на момент t_k при реализации второго случая. Получаемое в итоге расписание $\rho^{**}(t, Q)$ состоит из $n+r$ элементов, где r — число выполненных при его синтезе расщеплений (отметим, что полученные в результате расщеплений объекты могут подвергаться дальнейшим расщеплениям); $K(\rho^{**}(t, Q))$ — соответствующая вершине (t, Q) нижняя оценка LB.

Выбор вершины для очередного ветвления можно выполнять любым из стандартных способов. В выполненной авторами реализации очередное ветвление выполняется в открытой вершине наибольшего ранга (ранг вершины – число дуг в пути от корня дерева до рассматриваемой вершины).

Основанную на схеме ветвей и границ технологию решения задачи 1 обозначаем как алгоритм **B**.

Для сравнения быстродействия алгоритмов **A** и **B** были выполнены вычислительные эксперименты на тестовых наборах данных; при этом для каждого фиксированного значения n от 10 до 19 осуществлялся расчет для ста потоков O_n; параметры объектов в потоках задавались случайным образом по нормальному закону распределения из интервалов, представленных в таблице 1.

Таблица 1.

$t_i, i = \overline{1,n}$	$w_i = +1$			$w_i = -1$		
	a_i	τ_i	v_i	a_i	τ_i	v_i
$[t_i - 1, t_i - 1 + 10]$	$[7, 15]$	$[8, 20]$	$[V^*/10, V^*/2]$	$[1, 7]$	$[1, 5]$	$[V^*/20, V^*/4]$

Средние продолжительности решения задачи алгоритмами **A** и **B** в зависимости от количества объектов в потоке приведены в Таблице 2 с точностью до тысячной доли секунды.

Таблица 2.

n	**A**	**B**	n	**A**	**B**
10	0.013	0.000	15	208.760	0.502
11	0.059	0.001	16	832.329	1.560
12	0.466	0.006	17	-	9.542
13	2.124	0.028	18	-	43.459
14	18.678	0.107	19	-	478.901

5. Варианты сокращения продолжительности синтеза расписаний обслуживания

5.1. Первый вариант предусматривает решение задачи 1 комбинированным методом, использующим соотношения динамического программирования и оценки, свойственные методу ветвей и границ. При этом реализуется модификация алгоритма **A**, в которой для вновь получаемых состояний периодически дополнительно подсчитываются оценки UB и LB. Наименьшая из получаемых в процессе вычислений оценок UB именуется рекордом (начальное значение рекорда полагается равным $+\infty$; в процессе счета оно убывает). Как очевидно, раскрытие состояний, оценки LB которых превышают имеющееся значение рекорда, нецелесообразно. Уменьшая число раскрываемых состояний, мы сокращаем объем выполняемых вычислений.

В модифицированном алгоритме подсчет оценок выполняется после каждого p-го по порядку выполнения этапа 3 алгоритма **A**, где p такое, что при делении p на заданную константу k мы получаем в остатке число, меньшее m (k и m — параметры алгоритма, $m < k$). При этом переходим не к этапу 4 алгоритма **A**, а к дополнительному этапу 3^*.

Этап 3^*: а) для каждого нефинального состояния — пары (t, Q), присутствующей в записях списка G_3 в качестве четвертой-пятой компонент, выполняем процедуру получения верхней и нижней оценок $K(\rho^*(t, Q)))$ и $K(\rho^{**}(t, Q))$ соответственно; б) если минимальная из полученных при реализации данного этапа верхних оценок оказалась меньше рекорда R, то эта оценка принимается в качестве нового значения R; в) из списка G_3 изымаются записи $[t^*, Q^*, \alpha^*, t, Q, K(t, Q)]$ такие, что $K(\rho^{**}(t, Q)) > R$.

Далее выполняется переход к этапу 4.

5.2. Другой способ сокращения продолжительности синтеза расписаний обслуживания состоит в переходе к отдельным конкретизациям рассматриваемой модели (аналогично тому, как это для более простой задачи было сделано в [9]).

Два объекта назовем однотипными, если они принадлежат одному подпотоку, имеют одинаковую вместимость, одну и ту же нормативную длительность обслуживания и равные штрафы за единицу времени простоя.

Задачей $1^{p,q}$ назовем частный случай задачи 1, получаемый в предположении, что число типов объектов подпотоков O^+ и O^- не превышают заданных констант p и q соответственно (в имеющихся приложениях константы p и q — достаточно малы).

Будем считать, что типы объектов пронумерованы: объекты подпотока O^+ относятся к типам $1, 2, \ldots, p$; объекты подпотока O^- относятся к типам $p + 1, p + 2, \ldots, p + q$.

Любое множество объектов U будем характеризовать $(p + q)$-мерным вектором $D(U) = (d_1, d_2, \ldots, d_{p+q})$, где d_i — число принадлежащих множеству U объектов i-го типа, $i = 1, 2, \ldots, p + q$. Множества объектов U_1 и U_2 именуем равнозначными, если $D(U_1) = D(U_2)$. Отметим, что для введенных при рассмотрении задачи 1 частных задач $Z(t, Q)$ и оптимальных значений их критериев имеет место следующее: в случае равнозначности множеств Q_1 и Q_2 величины $K(t, Q_1)$ и $K(t, Q_2)$ совпадают. Благодаря отмеченному, функцию $K^*(t, W)$, где W — $(p+q)$-мерный вектор с целыми нотрицательными координатами, определяем следующим образом: $K^*(t, W) = K(t, Q)$, где Q — любое множество объектов такое, что $D(Q) = W$. Таким образом, $K^*(t, W)$ —

минимально возможный суммарный штраф по всем прибывшим объектам, завершенным обслуживанием не позднее момента t при условии, что множество прибывших, но не обслуженных по состоянию на этот момент времени объектов характеризуется вектором W.

Записанное выше основное рекуррентное соотношение (6) для вычисления значений функции $K^*(t, W)$ модифицируется очевидным образом, а общее количество рассматриваемых векторов W, являющихся аргументами этой функции, оценивается сверху величиной n^{p+q-1}.

В естественном для рассматриваемого класса задач предположении $t_n \leqslant C$, где C — константа, не зависящая от n, получаем, что основанный на соотношениях динамического программирования алгоритм решения задачи $1^{p,q}$ для фиксированных параметров p и q имеет полиномиально зависящую от числа подлежащих обслуживанию объектов оценку вычислительной сложности.

В приложениях достаточно часто при рассмотрении задачи $1^{p,q}$ имеется дополнительная информация, ограничивающая число ожидающих обслуживания объектов. Если считать, что по состоянию на любой момент принятия решения количество объектов, находящихся в системе и ожидающих обслуживания, не может превысить натуральную константу z, то общее количество рассматриваемых векторов W, являющихся аргументами функции $K^*(t, W)$, оценивается сверху величиной z^{p+q-1}, т.е. не зависящей от n константой. Число требуемых для решения задачи 1 вычислений значений функции $K^*(t, W)$ становится линейно зависящим от n, а основанный на принципе динамического программирования решающий алгоритм приобретает квадратично зависящую от n оценку вычислительной сложности.

6. Заключение

В работе описана модель одностадийного обслуживания бинарного потока объектов процессором с накопительно-расходным компонентом. Сформулирована задача синтеза расписания обслуживания, минимизирующего суммарный штраф по всем объектам; рассмотрены вопросы вычислительной сложности; на основе концепций динамического программирования, ветвей и границ, а также их совместной реализации построены решающие алгоритмы. Несмотря на полученные результаты об *NP*-трудности изучаемой задачи, выполненные на реальных данных расчеты потребовали вполне приемлемого расхода времени. Так, например, для каждой из возникших на практике задач из 17–19 объектов синтез оптимального расписания выполнялся на персональном компьютере (процессор Intel Core 2 Duo, 3,16 ГГц, память 4 Гб) с расходом времени в пределах 8–13 минут.

ПРИЛОЖЕНИЕ

Доказательства теорем 1 и 2 заключаются в выполнении полиномиальных сведений *NP*-полной в сильном смысле задачи «3-разбиение» [5] к проблемам, сформулированным в этих теоремах.

Задача «3-разбиение» состоит в следующем. Имеется конечное множество натуральных чисел $B = \{b_1, b_2, \ldots, b_{3n}\}$ такое, что каждое b_i принадлежит интервалу $(T/4, T/2)$ и $\sum_{i=1}^{3n} b_i = nT$; здесь T — некоторое натуральное число. Спрашивается, можно ли множество B разбить на n попарно непересекающихся подмножеств

$B_1, B_2, \ldots, B_n$ так, чтобы суммы чисел, входящих в каждое подмножество, совпадали. Отметим, что в случае положительного решения задачи в каждом из подмножеств B_i, $i = \overline{1,n}$ оказывается три числа, сумма которых равна T.

Доказательство теоремы 1. По исходным данным задачи «3-разбиение» модель обслуживания $\boldsymbol{M}_0^*$ строим следующим образом.

Совокупность O^+ состоит из объектов $o_1, o_2, \ldots, o_n$; совокупность O^- — из объектов $o_{n+1}, o_{n+2}, \ldots, o_{4n}$ (объекты совокупности O^+ предназначены для пополнения резервуара, объекты совокупности O^- — для заполнения из резервуара); все объекты готовы к обслуживанию от момента времени $t = 0$. Объем резервуара равен T, в момент времени $t = 0$ он полон, т.е. $V(0) = T$. Вместимость каждого объекта из совокупности O^+ полагается равной T; объект o_{n+i} из совокупности O^- имеет вместимость b_i, $i = 1, 2, \ldots, 3n$. Длительности обслуживания объектов и штрафы за единицу времени роли не играют.

Так как вместимость каждого объекта из совокупности O^+ равна T, обслуживание каждого следующего объекта из O^+ можно выполнять только тогда, когда заполнение резервуара оказалось нулевым; в результате обслуживания объекта из O^+ заполнение резервуара становится предельно возможным, т.е. равным T. От момента времени $t = 0$ до момента начала обслуживания первого объекта из O^+ должно быть обслужено подмножество объектов из совокупности O^-, определяемое некоторой совокупностью индексов $M(1)$ такое, что $\sum_{j \in M(1)} b_j = T$. Далее между каждым $(i-1)$-м и i-м объектом совокупности O^+(можно считать их пронумерованными в порядке обслуживания) должно быть обслужено подмножество объектов из совокупности O^-, определяемое некоторой совокупностью индексов $M(i)$ такое, что $\sum_{j \in M(i)} b_j = T$, $i = 2, 3, \ldots, n$. Получаем, что допустимое расписание обслуживания можно построить тогда и только тогда, когда исходная задача «3-разбиение» имеет положительное решение. Описанное сведение реализуется во времени, полиномиально зависящем от объема информации по исходной задаче, им не порождаются числа большие, чем T. Теорема доказана.

Доказательство теоремы 2. По исходным данным задачи «3-разбиение» модель обслуживания $\boldsymbol{M}_0^{**}$ строим следующим образом.

Совокупность O^+ состоит из объектов $o_1, o_2, \ldots, o_n$; совокупность O^- — из объектов $o_{n+1}, o_{n+2}, \ldots, o_{4n}$ (объекты совокупности O^+ предназначены для пополнения резервуара, объекты совокупности O^- — для заполнения из резервуара). Все объекты готовы к обслуживанию от момента времени $t = 0$. Объем резервуара V^* превосходит $2T$, в момент времени $t = 0$ он полон, т.е. $V(0) = V^*$. Вместимость каждого объекта из совокупности O^+ полагается равной T; объект O_{n+i} из совокупности O^- имеет вместимость b_i, $i = 1, 2, \ldots, 3n$. Далее полагаем: $a_1 = a_2 = \ldots = a_n = 1$; $a_{n+i} = 0$, $i = 1, 2, \ldots, 3n$; $\tau_1 = \tau_2 = \ldots = \tau_n = 1$; $\tau_{n+i} = b_i$, $i = 1, 2, \ldots, 3n$.

В отличие от модели $\boldsymbol{M}_0^*$, исходные данные модели $\boldsymbol{M}_0^{**}$ удовлетворяют не только условию (1), но и неравенствам (2). Поэтому в модели $\boldsymbol{M}_0^{**}$ множество допустимых расписаний непусто.

С целью получения нижней оценки оптимального значения критерия задачи 1_0, возникающей в рамках модели $\boldsymbol{M}_0^{**}$, допускаем расщепление объектов подпотока O^-. Тем самым расширяется множество решений, оптимальное значение критерия в новой задаче, назовем ее задачей $1_0'$, будет не больше оптимального значения критерия задачи 1_0. Произвольный объект o_{n+j}, $j \in \{1, 2, \ldots, 3n\}$, может быть расщеплен на два объекта, $o_{n+j[1]}$ и $o_{n+j[2]}$. При этом $\tau_{n+j[1]}$ и $\tau_{n+j[2]}$ полагаются равными $\xi_j \tau_{n+j}$ и $(1-\xi_j)\tau_{n+j}$ соответственно, а $b_{n+j[1]}$ и $b_{n+j[2]}$ равными $\xi_j b_{n+j}$ и $(1-\xi_j)b_{n+j}$ соответственно, здесь ξ_j — принадлежащая числовому интервалу $(0,1)$ константа. Задача $1'_0$ решается очевидным образом. В связи с тем, что вместимость каждого объекта из совокупности O^+(а штрафы налагаются только по объектам из O^+) равна T, обслуживание каждого следующего объекта из O^+ необходимо начинать когда заполнение резервуара оказывается равным $V^* - T$. Поэтому от момента времени $t = 0$ обслуживаются вплоть до достижения заполнения резервуара $V^* - T$ любые объекты множества O^- (возможно расщепление последнего из них). Затем обслуживается любой объект из совокупности O^+. Далее между $(i-1)$-м и i-м объектом совокупности O^+, $i = 2, 3, \ldots, n$ должно быть обслужено (с возможным расщеплением) подмножество объектов из совокупности O^-, определяемое некоторой совокупностью индексов $M(i)$ такое, что $\sum_{j \in Mi} b_j = T$.

Изложенная организация процесса обслуживания обеспечивает величину суммарного штрафа $\mathbf{S} = (T+1) + (2T+2) + \ldots + (nT+n) = n(n+1)(T+1)/2$; $\mathbf{S}$ — нижняя оценка оптимального значения критерия в задаче $1'_0$, где расщепление предметов невозможно. Указанная оценка достижима, оптимальное значение критерия в задаче $1'_0$, с нерасщепляемыми предметами равно $\mathbf{S}$ тогда и только тогда, когда имеющееся множество натуральных чисел $B = \{b_1, b_2, \ldots, b_{3n}\}$ можно разбить на n попарно непересекающихся подмножеств $B_1, B_2, \ldots, B_n$ так, чтобы суммы чисел, входящих в каждое подмножество, совпадали. Описанное сведение реализуется во времени, полиномиально зависящем от объема информации по исходной задаче, им не порождаются числа, большие, чем T. Теорема доказана.

СПИСОК ЛИТЕРАТУРЫ

1. Северный завоз / Материал из Википедии — свободной энциклопедии. URL: http://ru.wikipedia.org/wiki/Северный_завоз (дата обращения: 31.05.2013).

2. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы. М.: Наука, 1984. 382 с.

3. *Танаев В.С., Сотсков Ю.Н., Струсевич В.А.* Теория расписаний. Многостадийные системы. М.: Наука, 1989. 328 с.

4. *Танаев В.С., Шкурба В.В.* Введение в теорию расписаний. М.: Наука, 1975. 256 с.

5. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.

6. *Pinedo, M.L.* Scheduling. Theory, Algorithms, and Systems. Springer, 2008. 671 p.

7. *Корбут А.А., Финкельштейн Ю.Ю.* Дискретное программирование. М.: Наука, 1969. 368 с.

8. *Сигал И.Х., Иванова А.П.* Введение в прикладное дискретное программирование. М.: Наука, 2007. 237 с.

9. *Коган Д.И., Федосенко Ю.С.* Задача диспетчеризации: анализ вычислительной сложности и полиномиально разрешимые подклассы // Дискретная математика. 1966. Т. 8. Вып. 3. С.135-147.

Коган Д.И., *Московский государственный университет приборостроения и информатики, кафедра Прикладной математики и информатики, профессор, Москва*, `kdi_41@mail.ru`

Куимова А.С., *Волжская государственная академия водного транспорта, кафедра Информатики, систем управления и телекоммуникаций, аспирант, Нижний Новгород*, `anastasia.kuimova@gmail.com`

Федосенко Ю.С., *Волжская государственная академия водного транспорта, кафедра Информатики, систем управления и телекоммуникаций, заведующий кафедрой, Нижний Новгород*, `fds@aqua.sci-nnov.ru`