ИНФОРМАТИКА

2006 **январь-март** **№ 1**

УДК 519.8

А.А. Тарасевич

МИНИМИЗАЦИЯ СУММАРНОГО ШТРАФА ОБСЛУЖИВАНИЯ НЕОДНОВРЕМЕННО ПОСТУПАЮЩИХ ТРЕБОВАНИЙ С ОБЩИМ ДИРЕКТИВНЫМ СРОКОМ НА ОСНОВЕ ПОИСКА С ЗАПРЕТАМИ

Предлагается эвристический алгоритм, использующий метод поиска с запретами для решения задачи минимизации суммарного штрафа за опережение и запаздывание обслуживания требований и назначение общего директивного срока в условиях неодновременного поступления требований в системах с одним прибором.

Введение

При проектировании и управлении в промышленности и сфере обслуживания возникают задачи определения наилучшей последовательности выполнения работ с директивными сроками завершения, которые не заданы изначально, а назначаются таким образом, чтобы минимизировать критерий оптимальности расписания [1–3].

Рассматривается задача составления расписания, в которой n требований должны быть обслужены одним прибором в условиях назначения общего директивного срока d. Каждое требование j, $j = 1, 2, \ldots, n$, становится доступным (готовым к обслуживанию) в момент времени r_j (момент поступления) и должно обрабатываться p_j (длительность обслуживания) единиц времени. Пусть S_j и C_j – время начала и окончания обслуживания требования j в некотором расписании s. Обозначим через $E_j = \max\{0, d - C_j\}$ и $T_j = \max\{0, C_j - d\}$ соответственно опережение и запаздывание в обслуживании требования j относительно директивного срока.

Необходимо минимизировать функцию $f(d, s) = \sum_{j=1}^{n} (\alpha E_j + \beta T_j + \gamma d)$, где $\alpha > 0$, $\beta > 0$ и $\gamma > 0$ – штраф за единицу опережения, запаздывания и директивного срока соответственно. Иными словами, требуется найти оптимальное расписание s^* и оптимальный директивный срок d^*. Необходимо отметить, что значение $\beta = 0$ ведет к $d^* = 0$ и к оптимальности любого допустимого расписания. Случаи задачи с $\beta \le \gamma$ и $\alpha = 0$ эквивалентны NP-трудной в сильном смысле задаче минимизации суммы моментов завершения обслуживания в условии неодновременного поступления требований [4] (задача $1|r_j|\sum C_j$ по схеме [5]), которая хорошо изучена. Поэтому далее рассматривается случай, когда $\beta > \gamma$. Расширяя стандартную схему для обозначения задач составления расписаний [5], введем следующее обозначение для рассматриваемой задачи: $1|r_j, d_j := d|\sum(\alpha E_j + \beta T_j + \gamma d)$.

Пэнвалкар, Смит и Сайдманн рассмотрели случай задачи в условиях одновременного поступления требований ($r_j = 0$) [6]. Они доказали, что оптимальный общий директивный срок совпадает с моментом завершения обслуживания требования $[K]$ и предложили алгоритм для решения такой задачи за полиномиальное время. Здесь $[j]$ обозначает требование, стоящее j-м в последовательности, а K – наименьшее целое число, большее либо равное $n(\beta - \gamma)/(\alpha + \beta)$. Ченг, Чен и Шахлевич рассмотрели задачу с неодновременным поступлением требований [7]. Они предложили алгоритмы для решения задачи за полиномиальное время в случаях, когда длительности обслуживания одинаковы или последовательность требований известна. Для решения задачи в общем случае предложен эвристический алгоритм.

1. Эвристический алгоритм CCS

Эвристический алгоритм CCS [7] для решения задачи $1|r_j, d_j := d|\sum(\alpha E_j + \beta T_j + \gamma d)$ состоит из двух частей (алгоритмов 1 и 2).

Алгоритм 1 (Пэнвалкара, Смита и Сайдманна) [6] строит последовательность требований σ^*, оптимальную для задачи с нулевыми моментами поступления.

1. Перенумеруем требования в порядке неубывания их длительностей обслуживания.

2. Вычислим $K := \lceil n(\beta - \gamma)/(\alpha + \beta) \rceil$. Здесь $\lceil x \rceil$ – наименьшее целое число, большее либо равное x.

3. Присвоим метку каждой позиции *i* последовательности согласно следующей формуле:

$$\lambda_i = \begin{cases} n\gamma + (i-1)\alpha, & 1 \le i \le K; \\ (n+1-i)\beta, & K+1 \le i \le n. \end{cases}$$

4. Проранжируем позиционные метки λ_i в порядке убывания их величины. Таким образом, наибольшей метке λ_i присвоен ранг 1, а наименьшей – ранг *n*.

5. Построим последовательность $\sigma^* = ([1], \ldots, [i], \ldots, [n])$, где требование *j* ставится в позицию *i*, если *j* – ранг метки λ_i.

Алгоритм 2 строит расписание, оптимальное для заданной последовательности σ требований.

1. Определим моменты начала обслуживания требований по следующим формулам: $S_{[1]}(\sigma) = r_{[1]}$, $S_{[i]}(\sigma) = \max\{r_{[i]}, C_{[i-1]}(\sigma)\}$, где $[i]$ – требование в *i*-й позиции последовательности σ.

2. Вычислим $u := \lceil n(\beta - \gamma)/\beta \rceil$.

3. Сдвинем требования в позициях $1, 2, \ldots, u-1$ к требованию, стоящему в позиции *u*, таким образом, чтобы между ними не было прерываний в обслуживании.

4. Вычислим $K := \lceil n(\beta - \gamma)/(\alpha + \beta) \rceil$ и зададим $d_h^* := C_{[K]}$.

Применяя алгоритм 2 для последовательности $\sigma = \sigma*$, получаем расписание s_h^* и директивный срок d_h^* с учетом моментов поступления требований. Это расписание оптимально для заданной последовательности σ^*, но не оптимально, в общем случае, для задачи $1|r_j, d_j := d|\sum(\alpha E_j + \beta T_j + \gamma d)$.

Сочетание алгоритмов 1 и 2 назовем эвристикой CCS. Ченг, Чен и Шахлевич приводят следующую оценку точности эвристики CCS: $f(s_h^*, d_h^*)/f(s*, d*) \le 1 + R$ [7]. Здесь $s*$ – оптимальное расписание обслуживания требований, $d*$ – оптимальный общий директивный срок, $R = \max_j\{r_j\}$.

2. TS-эвристика

Методы локального поиска решают задачу, переходя от одного (текущего) решения к другому (соседнему). Их эффективность зависит от правил поиска соседнего решения и от правила принятия этого решения в качестве текущего после сравнения значений критерия оптимальности. Методы локального поиска в общем случае не приводят к глобальному минимуму, так как большинство из них принимает в качестве текущих только те соседние решения, которые улучшают критерий оптимальности.

Метод поиска с запретами используется в оптимизационных задачах для ускорения поиска решения [8, 9]. Соседнее решение ищется в некоторой окрестности текущего решения. При

попытке перехода к соседнему решению, ухудшающему текущее более чем на некоторый заданный порог (может быть, нулевой), это соседнее решение помещается в запретный список с тем, чтобы исключить его из рассмотрения в случае, если будет попытка перейти к нему снова. Затем повторяется попытка перехода от нового текущего решения к наилучшему в его окрестности. Поиск прекращается в случае, если в окрестности все решения, кроме занесенных в запретный список, хуже текущего более чем на величину порога. Текущее решение и является результатом поиска. Для улучшения результатов можно повторять поиск несколько раз. На скорость и качество решения могут влиять длина списка запретных решений, а также размер окрестности решения, по которой осуществляется поиск наилучшего соседнего решения.

Ниже приведены основные шаги решения задачи предлагаемой эвристикой.

1. Построим начальную последовательность σ требований путем упорядочивания их в порядке неубывания моментов поступления. Возьмем решение $S^o = (s^o, d^o, f^o)$ в качестве наилучшего из найденных. Здесь f^o – значение функции штрафов и d^o – общий директивный срок для расписания s^o, полученного при помощи алгоритма 2 для последовательности σ.

2. Возьмем решение S^o в качестве текущего. Обозначим через $f*$ значение функции штрафов для текущего решения. Выполним следующие действия n раз: построим последовательность σ' для соседнего решения путем перестановки двух случайно выбранных требований в последовательности текущего решения. Проверим наличие этой последовательности в списке запретных решений и, в случае обнаружения, перейдем к построению следующей последовательности σ'. Если же эта последовательность не обнаружена в списке запретных решений, то применим алгоритм 2 к последовательности σ', чтобы получить соседнее решение со значением f' целевой функции. Если соседнее решение не хуже текущего с точностью Δ, т. е. $f' \le f* + \Delta$, тогда примем его в качестве текущего. Если текущее решение лучше решения S^o, т. е. $f* < f^o$, примем его в качестве решения S^o. Если соседнее решение хуже текущего с точностью Δ (т. е. $f' > f* + \Delta$), помещаем его в список запретных решений.

3. Если цикл 2 повторялся n^2 раз, завершим поиск: решение S^o – наилучшее из найденных – результат работы эвристики. В противном случае цикл 2 повторим заново.

Следующие особенности такого подхода позволили улучшить результат и значительно сократить время выполнения алгоритма:

– каждые n переходов в качестве текущего решения принимается текущее наилучшее.

– порог $\Delta = 0$ обеспечил наибольшую эффективность решения задачи.

Далее приведено детальное описание алгоритма 3, реализующего данный подход к решению задачи:

1. Зададим $t := t_0$.

2. Построим последовательность $\sigma = ([1], [2], \ldots, [n])$, упорядочив требования в порядке неубывания моментов их поступления, и получим оптимальные для этой последовательности расписание $s*(\sigma)$ и директивный срок $d*(\sigma)$, используя алгоритм 2.

3. Вычислим величину целевой функции для расписания $s*(\sigma)$ и директивного срока $d*(\sigma)$:

$$f* = f(d*, s*) = \sum_{j=1}^{n} (\alpha E_j + \beta T_j + \gamma d).$$

Пусть $s*(\sigma)$, директивный срок $d*(\sigma)$ и функция штрафов $f*$ являются текущим решением. Запомним это текущее решение как S^o $(s^o, d^o, f^o) := (s*, d*, f*)$.

4. *СчетчикЦиклов* := 0.

5. $i := 0$; *СчетчикЦиклов* := *СчетчикЦиклов* + 1.

6. $i := i + 1$.

7. Поменяем местами требование в позиции *i* и требование в случайно выбранной позиции *j* в последовательности σ и получим соседнее решение для новой последовательности σ' ($s^*(\sigma')$, $d^*(\sigma')$, f').

7.1. Если текущая последовательность σ' есть в списке запретных решений, тогда перейдем к шагу 6.

7.2. Определим $d^*(\sigma')$ и $s^*(\sigma')$, используя алгоритм 2 для σ'.

7.3. Вычислим $f' = f(d^*(\sigma'), s^*(\sigma'))$.

8. Сравним соседнее решение с текущим.

8.1. Если $f' - \Delta \le f^*$, тогда текущее решение := соседнее решение.

8.2. Если $f' - \Delta < f^o$, тогда S^o := соседнее решение.

8.3. Если $f' - \Delta > f^*$, поместим соседнее решение (последовательность σ' и $f' = f(d^*(\sigma'), s^*(\sigma'))$) в список запретных решений.

9. Если $i < n$, перейдем к шагу 6.

10. Текущее решение := наилучшее решение.

11. Проверим условия завершения работы:

11.1. Если *СчетчикЦиклов* $= n^2$, перейдем к шагу 13.

11.2. Если *СчетчикЦиклов* $= n$, построим произвольную последовательность требований и расписание для нее.

12. Перейдем к шагу 5.

13. Текущее решение является наилучшим решением, полученным эвристикой. Стоп.

Примечание: СчетчикЦиклов – счетчик количества циклов из n итераций (шаги 6–9).

Шаги 4–12 повторяются n^2 раз, шаги 5–9 – n раз и на шаге 7.2 выполняется алгоритм 2 сложности $O(n)$. Таким образом, временная сложность алгоритма 3 составляет $O(n^4)$.

3. Численный эксперимент

Алгоритм 3 был реализован с использованием Microsoft Visual C++ 6.0. Все вычисления производились на компьютере с процессором Athlon XP (1800 МГц). Естественно, что предложенная эвристика получала решения, лучшие, чем те, которые получает алгоритм CCS для одних и тех же экземпляров задач. Для того чтобы сравнить эти эвристики, определялись отношения f_{ts} / f_{ccs}, где f_{ts} и f_{ccs} – значения функций штрафов, получаемые предложенной эвристикой и CCS-алгоритмом соответственно для одного и того же экземпляра задачи. Значения средних оценок f_{ts} / f_{ccs} получены в пределах от 0,495 до 0,966. Наибольшие оценки (1 и 1,02) были получены для нескольких экземпляров задач со значениями $k = \{0{,}01;\ 0{,}03;\ 0{,}05\}$ и $n = 20$. Для остальных сочетаний *n* и *k* $f_{ts} / f_{ccs} < 1$ для всех экземпляров задач.

Необходимо отметить, что результаты для задач с малыми значениями *k*, полученные TS-эвристикой, близки к результатам, полученным CCS-алгоритмом. Это объясняется тем, что в CCS-алгоритме последовательность требований строится для упрощенной задачи с нулевыми моментами поступления. Поэтому, когда моменты поступления r_j распределены в малых пределах, решения, получаемые этим алгоритмом, близки к оптимальным.

Экземпляры задач были сгенерированы с использованием генератора тестовых примеров для задач теории расписаний с одним прибором, предложенного Бискупом и Фельдманом [10].

При генерации экземпляров задач использовались следующие четыре параметра:

n – количество требований ($n = 20;\ 50;\ 100;\ 1000$);

$p_{\max}$ – верхний предел для длительностей обслуживания требований ($p_{\max} = 20;\ 50;\ 100$);

$r_{\max} = knp_{\max}$ – верхний предел для генерируемых случайно моментов поступления требований (k = 0,01; 0,03; 0,05; 0,1; 0,2; 0,3; 0,5; 1; 1,5; 2; 1/n);

$\mu_{\max}$ – верхний предел для значений α, β, γ ($\mu_{\max}$ = 10; 20; 30).

Для каждого экземпляра задач значения α, β, γ выбирались случайным образом в диапазоне $[1, \mu_{\max}]$ с учетом условия $\beta > \gamma$; p_j и r_j выбирались из диапазонов $[1, p_{\max}]$ и $[1, r_{\max}]$ соответственно. Эксперименты показали, что картина результата зависит от $r_{\max}$ и практически не зависит от $\mu_{\max}$ и $p_{\max}$ (кроме зависимости от $p_{\max}$ через $r_{\max}$).

Из табл. 1 видно, что почти для всех экземпляров задач значения целевой функции, полученные TS-эвристикой, оказались меньше значений, полученных алгоритмом CCS. В среднем все значения лежат в диапазоне от 0,495 до 0,966.

Таблица 1

Значения f_{ts} / f_{ccs} для каждого значения n в зависимости от k

k	n=20	n=50	n=100	N=1000
1/n	0,887	0,919	0,926	0,966
0,01	0,932	0,927	0,931	0,958
0,03	0,908	0,897	0,895	0,892
0,05	0,888	0,863	0,860	0,846
0,1	0,829	0,797	0,786	0,787
0,15	0,785	0,748	0,734	0,723
0,2	0,748	0,707	0,685	0,672
0,25	0,724	0,673	0,655	0,711
0,3	0,702	0,645	0,631	0,586
0,5	0,663	0,606	0,583	0,495
0,7	0,656	0,610	0,586	0,576
1,0	0,676	0,633	0,625	0,576
1,5	0,704	0,670	0,660	0,645
2,0	0,721	0,694	0,682	0,670

Таблица 2

Время, затраченное на решение одного экземпляра задачи для каждого значения n в зависимости от k

k	n=20	n=50	n=100	n=1000
1/n	0	0,003	0,059	2292
0,01	0	0,003	0,061	2846
0,03	0	0,003	0,042	3646
0,05	0	0,003	0,040	3900
0,1	0	0,002	0,028	4661
0,15	0	0,002	0,027	5709
0,2	0	0,002	0,020	11837
0,25	0	0,002	0,018	6968
0,3	0	0,002	0,017	6834
0,5	0	0,001	0,009	5467
0,7	0	0,001	0,007	3887
1,0	0	0,001	0,007	3304
1,5	0	0,001	0,009	3324
2,0	0	0,001	0,009	2146

Время, затраченное TS-эвристикой для решения одного экземпляра задач (табл. 2), превышает время, затрачиваемое CCS-алгоритмом для того же экземпляра задач (оно составляет менее 0,001 с для $n < 100$ и менее 15 с для $n < 100$).

Для экземпляров задач с $n = 1000$ были взяты меньшие значения $p_{\max}$ (6, 16, 33) во избежание случаев превышения разрядности чисел в памяти компьютера.

Заключение

Для NP-трудной задачи минимизации штрафа за опережение и запаздывание обслуживания требований одним прибором при построении расписания с назначением общего директивного срока предложен эвристический алгоритм решения, который использует метод поиска с запретами и позволяет получать результаты, лучшие, нежели полученные с помощью предложенного ранее приближенного метода [7]. Время решения задач с размерностью до 100 требований не превышает 1 с.

Данная работа была выполнена при частичной поддержке INTAS проекта 03-51-5501.

Список литературы

1. Гордон В.С., Смотряев В.Н., Тарасевич А.А. Построение оптимальных расписаний при назначении директивных сроков // Информатика. – 2004. – № 1. – С. 17–27.

2. Гордон В.С., Смотряев В.Н., Тарасевич А.А. Задачи построения оптимальных расписаний с назначаемыми директивными сроками // Докл. Первого научного семинара «Танаевские чтения». – Мн.: ОИПИ НАН Беларуси, 2003. – С. 57–62.

3. Gordon V.S., Proth J.-M., Strusevich V.A. Scheduling with due date assignment // Handbook of scheduling: algorithms, models and complexity analysis. – Boca Raton: CRC Press, 2004. – Ch. 21. – P. 21-1 – 21-22.

4. Sequencing and scheduling: Algorithms and complexity / E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys // Logistics of Production and Inventory. Handbooks in Operations Research and Management Science. – Amsterdam: North-Holland, 1993. – Vol.4. – P. 445–522.

5. Lenstra J.K., Rinnooy Kan A.H.G., Brucker P. Complexity of machine scheduling problems // Annals of Discrete Mathematics. – 1977. – № 1. – P. 343–362.

6. Panwalkar S.S., Smith M.L., Seidmann A. Common due date assignment to minimize total penalty for the one machine scheduling problem // Operations Research. – 1982. – № 30. – P. 391–399.

7. Cheng T.C.E., Chen Z-L., Shakhlevich N.V. Common due date assignment and scheduling with ready times // Computers & Operations Research. – 2002. – № 29. – P. 1957–1967.

8. Hertz A., Taillard E., de Werra D. Tabu Search // Local Search in Combinatorial Optimization. – Wiley: Chichester, 1997. – P. 121–136.

9. Biskup D., Feldmann M. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates // Computers & Operations Research – 2001. – № 28. – P. 787–801.

Поступила 08.09.05

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: Tarasevich.Alexander@gmail.com

A.A. Tarasevich

MINIMIZING TOTAL PENALTY FOR SHEDULING JOBS WITH NON-ZERO RELEASE DATES UNDER A COMMON DUE DATE BASED ON TABU SEARCH

In the paper one can find a tabu search heuristic algorithm for scheduling jobs with non-zero release dates on a single machine to minimize the total penalty for early and tardy completion of jobs and assignment of a common due date.