2014январь-март № 1

УДК 519.95

Ф. Вернер[1], С.А. Кравченко[2], К. Хасани[3]

МИНИМИЗАЦИЯ СУММАРНОГО ВРЕМЕНИ ОБСЛУЖИВАНИЯ ДЛЯ СИСТЕМЫ С ДВУМЯ ПРИБОРАМИ И ОДНИМ СЕРВЕРОМ

Рассматривается задача минимизации суммарного времени обслуживания множества требований на множестве двух идентичных параллельных приборов. Перед обслуживанием требования необходима загрузка, которая осуществляется сервером. Известно, что задача NP-трудна в сильном смысле. В работе предлагаются две модели целочисленного линейного программирования и алгоритм имитации отжига (simulatedannealingalgorithm). Предложенные подходы тестируются на примерах задач, содержащих до 250 требований.

Введение

В работе рассматривается задача оптимального обслуживания n требований на множестве, состоящем из двух параллельных идентичных приборов. Для каждого требования j, j = 1, …, n, заранее известно время обслуживанияp_j. Обозначим через p_{max} максимальное значение для времени обслуживания, т.е. $p_{\max} = \max\{p_j \mid j = 1, \ldots, n\}$. Перед началом обслуживания требование необходимо загрузить на прибор M_q, q = 1,2, при помощи сервера (робота). Для загрузки требуется времяs_j, и загрузка может быть выполнена только в случае, если имеются свободный сервер и свободный прибор. Если же в текущий момент все приборы или сервер заняты, то загрузка нового требования не может быть начата. Предполагается, что время перевода сервера от одного прибора к другому пренебрежимо мало. Во время загрузки приборM_q так же, как и сервер, задействован в течение времени s_j. Любая загрузка выполняется сервером, который может грузить не более одного требования одновременно. Необходимо построить расписание, минимизирующее суммарное время обслуживания всех требований. С использованием общепринятых обозначений наша задача может быть записанакак $P2,S1||\sum C_j$. Данная задача является *NP*-трудной в сильном смысле, поскольку известно, что задача $P2,S1|s_j = s|\sum C_j$ тоже является *NP*-трудной в сильном смысле [1]. Известна вычислительная сложность для некоторых частных случаев рассматриваемой задачи. Задача $P2,S1|p_j = p|\sum C_j$ является *NP*-трудной в обычном смысле [2]. Известен полиномиальный алгоритм сложности $O(n\log n)$ для задачи $P2,S1|s_j = 1|\sum C_j$ [1]. Для задачи $P3,S1|s_j = 1|\sum C_j$ в [2] разработан полиномиальный алгоритм сложности $O(n^7)$. Для задачи $P,S1|s_j = 1|\sum C_j$ в [3] предложен алгоритм, который строит расписание с абсолютной погрешностью $\sum_{i=1}^{n}\tilde{C}_i - \sum_{i=1}^{n}C_i^* \le n'(m-2)$, где$m$ – число приборов, $n' = |\{i \mid p_i < m-1\}|$, C_i^* обозначает время завершения обслуживания требования i в оптимальном расписании, $\tilde{C}_i$ обозначает время завершения обслуживания требования i в построенном расписании. В работе [4] предложен приближенный алгоритм для задачи $P,S1||\sum w_jC_j$ с относительной погрешностью $\sum w_j\tilde{C}_j \big/ \sum w_jC_j^* \le (5-\frac{1}{m})$ и там же показано, что правило SPT позволяет строить расписания для задачи $P,S1|s_j = s|\sum C_j$ с гарантированной оценкой относительной погрешности 3/2. Авторам не известны публикации с эвристическими алгоритмами для задачи $P2,S1||\sum C_j$, однако известны результаты разработки эвристик для близких моделей. В работе [5] рассматривается задача построения расписания для обслуживания множества требований на множестве приборов разной производительности. При этом время загрузки требований зависит

от порядка обслуживания (sequencedependentsetup) и критерием является минимизация взвешенной суммы моментов завершения обслуживания требований. Несколько алгоритмов было предложено и протестировано. Максимальное количество требований и приборов, для которых применялись эвристики, равны соответственно 120 и 12. В работе [6] рассматривалась задача с идентичными приборами и требованиями, разделенными на семейства. Загрузчик использовался только в случае, если прибор, окончив обработку требования из одного семейства, должен обрабатывать требование из другого семейства. Критерием была минимизация взвешенной суммы моментов завершения обслуживания требований. Предложенные эвристики тестировались на примерах, в которых число семейств не превышало 8, количество требований не превышало 25 и количество приборов не превышало 5. Аналогичная предыдущей задача рассматривалась в работе [7], где было предложенонесколько алгоритмов типа ветвей и границ. Алгоритмы тестировались на примерах с максимальным количеством требований 25, максимальным количеством семейств 8 ис количеством приборов, не превышающим 5. В работе [8] рассматривалась задача с критерием $\sum C_j$ и одним сервером, однако не допускалось одновременное использование сервера и прибора при загрузке. Таким образом, задача моделировалась как двухстадийная задача гибридного типа с одинаковыми маршрутами обслуживания требований. При этом ставилось условие отсутствия задержек между стадиями при обслуживании. Для задачи была предложена математическая модель и рассматривались некоторые полиномиально разрешимые частные случаи. В данной работе исследуется задача $P2, S1||\sum C_j$ и предлагаются две модели целочисленного линейного программирования и один эвристический алгоритм имитации отжига. Тесты выполняются для примеров с количеством требований, не превышающим 250.

1. Моделирование задачи $P2, S1||\sum C_j$

Поскольку для построения расписания достаточно знать порядок загрузки требований, оптимальное расписание для задачи $P2, S1||\sum C_j$ можно искать в классе списочных расписаний, т.е. все требования выполняются по списку в порядке очереди. При этом очередное требование обслуживается таким образом, чтобы время его завершения было минимальным.

Пример. Рассмотрим пример с пятью требованиями:s_1=2, p_1=4,s_2=2, p_2=3,s_3=1, p_3 = 5,s_4=2, p_4=4,s_5=1, p_5=2. Предположим, где требования обслуживаются в соответствии со списком π_0 = (3, 1, 4, 2, 5).

Для данного случая требования 3–5 обслуживаются на приборе M_1 , а требования 1, 2 – на приборе M_2. Суммарное время обслуживания – 53.

Обозначим минимальное значение суммы моментов завершения обслуживания всех требований как $\sum C_j^*$. ОбозначимчерезL_j сумму s_j + p_j дляj = 1, …, n. Везде далее предполагаем, что требования упорядочены таким образом, что справедливо $L_1 \le \ldots \le L_n$.

Утверждение.*Для расписания, построенного в соответствии со списком* $\{1, \ldots, n\}$*, где выполняется* $L_1 \le \ldots \le L_n$*, справедливо неравенство* $\dfrac{\sum \tilde{C}_j}{\sum C_j^*} \le 2$.

Здесь $\sum \tilde{C}_j$ обозначает сумму моментов завершения обслуживания требований для расписания, соответствующего списку (1, …,n).

Доказательство. Легко видеть, что справедливо

$$\tilde{C}_j \le \sum_{k=1}^{j-1} s_k + \frac{1}{2}\sum_{k=1}^{j-1} p_k + L_j = \frac{1}{2}\sum_{k=1}^{j-1} s_k + \frac{1}{2}\sum_{k=1}^{j-1} L_k + L_j .$$

Поскольку $\frac{1}{2}\sum_{k=1}^{j-1} s_k + \frac{1}{2}L_j \le \frac{1}{2}\sum_{k=1}^{j} L_k \le C^*_{[j]}$ и $\frac{1}{2}\sum_{k=1}^{j-1} L_k + \frac{1}{2}L_j \le C^*_{[j]}$, получаем $\tilde{C}_j \le 2C^*_{[j]}$. Здесь $C^*_{[j]}$ обозначает время завершения обслуживания j-го (по порядку завершения обслуживания)требования в оптимальном расписании. При этом используется неравенство $\frac{1}{2}\sum_{k=1}^{j} L_k \le C^*_{[j]}$, которое следует непосредственно из $L_1 \le \ldots \le L_n$. ■

В дальнейшем используются две нижние границы для оценки $\sum C_j$. Первая нижняя граница LB_1 вычисляется по правилу $\sum_j C^1_j = LB_1$, где $C^1_j = L_j + L_{j-2} + \ldots + L_{j-2n}$, и все L_kс $k \le 0$ полагаем равными нулю. Вторая нижняя граница LB_2 вычисляется по правилу $\sum_j C^2_j = LB_2$, где $C^2_j = L_j + ss_{j-1} + ss_{j-2} + \ldots + ss_1$ и $ss_1, \ldots, ss_n -$ времена загрузки, перенумерованные в порядке неубывания их значений, т.е. $ss_1 \le \ldots \le ss_n$.

1.1. Модель M1

Модель M1 предлагается для нахождения оптимального расписания для задачи $P2, S1 \| \sum C_j$, причемоптимальное расписание ищется в классе списочных расписаний. Предположим, что имеется расписаниеs, соответствующее списку π. Обозначим символом ft_jвремя завершенияобслуживания требования j, а символом st_j– момент времени, соответствующий началу загрузки требования j. Кроме того, пусть mt_j обозначает момент времени, соответствующий окончанию загрузки требования j. Обозначим $O_{i,j}$ бинарную переменную, определяющую порядок требований i и j в списке π, т.е. $O_{i,j}$ = 1,если $st_i \le st_j$, а иначе $O_{i,j}$ = 0. Для бинарной переменной $p_{i,j}$полагаем $p_{i,j}$ = 1в случае $mt_i \le st_j \le ft_i$, т.е. если загрузка требованияj начинается после загрузки требования i, но до завершения его обслуживания. Минимизация $\sum C_j$ эквивалентна минимизации $\sum ft_j$. Поскольку $O_{i,j}$определяет порядок требований i и j в списке π, равенство$O_{i,j}$+ $O_{i,j}$= 1 должно выполняться. Неравенства $ft_j - st_j \ge s_j + p_j$ и $mt_j - st_j \ge s_j$ должны выполняться по определению. Поскольку имеется только два прибора, выполняется неравенство $\sum_{i=1}^{n} p_{i,j} \le 1$.

В дальнейшем нам понадобится большое число P. Для ускорения работы с моделью удобно взять $P = \frac{\sum_{i=1}^{n} p_i - p_{\max}}{2} + \sum_{i=1}^{n} s_i + p_{\max}$. При увеличении значения P время поиска оптимального решения увеличивается. Если требование jпредшествует требованию i в спискеπ, выполняется $st_j - mt_i \ge -P$. Однако если требование jследует после требования i в списке π, выполняется $st_j - mt_i \ge (1 - p_{i,j})\, p_i$; таким образом, если требование jне обслуживается одновременно с требованием i, выполняется $st_j - mt_i \ge p_i$. В итоге получаем неравенство $st_j - mt_i \ge (O_{i,j} - p_{i,j})\, p_i - O_{j,i} P$.

Итак, получаем модель

$$\sum_j ft_j \to \min;$$

$$O_{i,j} + O_{j,i} \underset{i \ne j}{=} 1;$$

$$ft_j - st_j \geq s_j + p_j;$$

$$mt_j - st_j \geq s_j;$$

$$\sum_{i=1}^{n} p_{i,j} \leq 1;$$

$$st_j - mt_i \geq (O_{i,j} - p_{i,j})\, p_i - O_{j,i} P;$$

$$\sum_j ft_j \geq LB.$$

Переменные $O_{i,j}$ и $p_{i,j}$ являются бинарными; переменные ft_i, st_iи mt_iявляются положительными; числа p_j и s_j известны заранее, i = 1, …, nиj = 1, …, n, а число Pявляется заданной константой; $LB = \max\{LB_1, LB_2\}$.

1.2. Модель M2

Модель M2 предлагается для поиска оптимального расписания в узком подмножестве, а именно когда требования из списка π должны обслуживаться с обязательной заменой прибора, т. е. рядом стоящие в списке π требования обслуживаются на разных приборах. Очевидно, что модель M2 не строит оптимальное расписание для задачи $P2, S1 \| \sum C_j$, но расписания, генерируемые такой моделью, оказываются близки к оптимальному расписанию. Опишем модель M2. Предположим, что расписание s задано спискомπ. Переменная $N_{i,j}$ определяет расположение требования j в списке π, т.е. $N_{i,j}$ = 1, если требование jнаходится в позиции i, а иначе $N_{i,j}$ = 0. Переменная ft_iобозначает время завершения обслуживания i-го требования из спискаπ,переменная st_i – время начала загрузки i-го требования из списка π,переменная mt_i–время завершения загрузки i-го требования из спискаπ. Минимизация $\sum C_j$ соответствует минимизации $\sum ft_j$. Поскольку каждой позиции из спискаπ соответствует только одно требование, должно выполняться равенство $\sum_j N_{i,j} = 1$. Поскольку каждое требование занимает некую позицию в списке π, выполняется $\sum_i N_{i,j} = 1$. Неравенство $ft_i - st_i \geq \sum_j N_{i,j}(s_j + p_j)$ выполняется, поскольку между моментом st_iи моментом ft_i должно обслужиться i-е требование из списка π. Аналогично, неравенство $mt_i - st_i \geq \sum_j N_{i,j} s_j$ справедливо, поскольку между моментами времени st_i и mt_iдолжно загрузитьсяi-е требование из списка π. Неравенство $st_i \geq mt_{i-1}$ выполняется, так как список π определяет порядок загрузки требований. Из-за чередования приборов должно выполняться неравенство $st_i \geq ft_{i-2}$.

Итак, модель M2 можно описать следующим образом:

$$\sum_i ft_i \to \min;$$

$$\sum_i N_{i,j} = 1;$$

$$\sum_j N_{i,j} = 1;$$

$$ft_i - st_i \geq \sum_j N_{i,j}(s_j + p_j);$$

$$mt_i - st_i \geq \sum_j N_{i,j} s_j;$$

$$st_i \geq mt_{i-1};$$

$$st_i \geq ft_{i-2}.$$

Переменные $N_{i,j}$ являются бинарными;ft_i, st_i и mt_i–положительные переменные; p_jи s_j – заранее известные константы; $i = 1, \ldots, n$ и $j = 1, \ldots, n$.

2. Алгоритм имитации отжига

Опишем алгоритм имитации отжига(SA) для задачи $P2,S1||\sum C_j$. На каждой итерации для заданного расписания случайным образом генерируется соседнее. В случае если значение критерия для сгенерированного соседа оказывается меньше заданного, соседнее расписание принимается за новое заданное расписание. В противном случае соседнее расписание принимается за новое заданное расписание с некоторой вероятностью.

В алгоритме SAдля генерации соседних расписаний используются следующие операции:

Перестановка: переставляются два случайно выбранные требования. Пусть для π_0= (3, 1, 4, 2, 5) из примера (см. разд. 1) две случайно выбранные позиции будут $a = 3$ и $b = 5$; применяя перестановку, получаем$Swap(\pi_0, a, b) = Swap(\pi_0, 3, 5) = (3, 1, 5, 2, 4)$.

*Перестановка соседних позиций:*переставляются два случайно выбранные соседние требования.Пусть для π_0= (3, 1, 4, 2, 5)две выбранные позиции будут $a = 3$ и $b = 4$. Получим$SwapAdj(\pi_0, a, b) = SwapAdj(\pi_0, 3, 4) = (3, 1, 2, 4, 5)$.

*Перестановка блока:*случайно выбранный блок, состоящий из ℓ требований, переставляется с другим случайно выбранным блоком, содержащим ℓ требований. Длина блока ℓ случайно выбирается из множества $\{2, 3, \ldots \left\lfloor \frac{n}{2} \right\rfloor\}$. После этого случайно определяется положение a первого требования в первом блоке и положение b первого требования во втором блоке. При этом$a + \ell \leq b \leq n - \ell + 1$. Для π_0= (3, 1, 4, 2, 5) из примера возможен выбор $\ell = 2$, $a = 2$ и $b = 4$. Тогда$SwapBlock(\pi_0, a, b, \ell) = SwapBlock(\pi_0, 2, 4, 2) = (3, 2, 5, 1, 4)$.

*Подстановка:*случайно выбранное требование удаляется с первоначальной позиции a и помещается в новую, случайно выбранную, позицию b. Для π_0= (3, 1, 4, 2, 5) из примера две случайно выбранные позиции $a = 2$ и $b = 4$ дают $Insert(\pi_0, a, b) = Insert(\pi_0, 2, 4) = (3, 4, 2, 1, 5)$.

*Подстановка блока:*случайно выбранный блок удаляется с позиции a и помещается в позицию b. Длина блока ℓ случайно выбирается из множества $\{2, 3, \ldots, n - b\}$. После выбора ℓ определяются позиции a и b, причем$b + \ell - 1 \leq n$. Для π_0= (3, 1, 4, 2, 5) из примера возможен выбор $\ell = 2$, $a = 2$ и $b = 4$. Тогда $InsertBlock(\pi_0, a, b, \ell) = InsertBlock(\pi_0, 2, 4, 2) = (3, 2, 5, 1, 4)$.

*Поворот блока:*поворачивается ℓ элементов последовательности. Для π_0= (3, 1, 4, 2, 5) из примера возможное значение $\ell = 3$ и $a = 2$. Тогда $ReverseBlock(\pi_0, a, \ell) = ReverseBlock(\pi_0, 2, 3) = (3, 2, 4, 1, 5)$.

*Подстановка блока и поворот:*после подстановки блока производится поворот. Дляπ_0 = = (3, 1, 4, 2, 5) изпримеравозможно $\ell = 2$, $a = 2$ и $b = 4$. Тогда $InsertReverseBlock(\pi_0, a, b, \ell) = InsertReverseBlock(\pi_0, 2, 4, 2) = (3, 2, 5, 4, 1)$.

На каждой итерации алгоритма имитации отжига используются все семь операций, генерируя семь соседних расписаний. Затемвыделяетсясоседнеерасписаниеснаименьшимзначением критерия. Полученное соседнее расписание принимается за новое по правилу, описанному выше.

Результаты тестирования нескольких схем тушения показали, что геометрическая схема слегка превосходит остальные,поэтому выбрали ее. Геометрическая схема тушения уменьшает температуру по правилу $T_k = \alpha T_{k-1}$, $k = 1,2,\ldots$, где $0 < \alpha < 1$. Было установлено, что начальная температура должна выбираться таким образом, чтобы в 25% случаев худшее расписание принималось за новое. Начальная температура в рассматриваемом алгоритме $T = 15$. Конечная температура должна быть достаточно низкой, чтобы худшие решения перестали заменять лучшие. Экспериментально было подобрано значение $\alpha = 0{,}999$. После каждой итерации температура в алгоритме изменяется. Фактически температура должна изменяться пропорционально отведенному времени, т.е. когда лимит времени *TL* исчерпан, температура

должна быть близка к 0. В соответствии с геометрической схемой тушения на последнем шаге алгоритма $T_N = \alpha^N T_0$. Получаем $N = \log_\alpha \frac{T_N}{T_0}$. Выбираем T_N = 0,0005, при таком выборе алгоритм содержит N = 10 304 итераций.

Начальное расписание генерируется случайным образом. Критерием остановки служит первое из возможных событий:

– исчерпано выделенное время;

– за последние 2000 итераций значение целевой функции не улучшалось;

– для целевой функции *BestSumC* выполняется *BestSumC* – *LB*< 1;

– текущаятемператураниже 0,0005.

В алгоритме SA*Rand*(0,1) обозначает некое число из интервала (0,1), случайно выбранное и равномерно распределенное.

```
BEGIN
   Генерируем начальное расписание X и вычисляем значение целевой функции SumC(π);
   BestSol :=π; BestSumC := SumC(π)
   T := начальная температура;
   WHILE (критерий остановки не выполнен) DO
      π':= лучший сосед среди сгенерированных соседей для расписания π;
      ΔC := SumC (π') – SumC (π);
      prob := Rand (0,1);
      IF ((ΔC ≤ 0) или (prob<e^(-ΔC/T))) THEN
         π := π';SumC(π) := SumC(π');
         IF (SumC(π) <BestSumC) THEN
            BestSumC :=SumC(π); BestSol := π;
         END IF
      END IF
   T :=Update (T);
   END WHILE
   OutputBestSolизначениеSumC;
END.
```

3. Результаты тестирования

Эффективность моделей M1, M2 и алгоритма SA тестировалась на данных, генерируемых подобно тому, как описано в [9]. Предельное время установлено равным (300/8)n с для примеров с $n \in \{8, 20, 50\}$ и 3600 сдля остальных примеров.

Для $n \in \{8, 20\}$ и для каждого $L \in \{0{,}1, 0{,}2, \ldots, 2\}$ значение s_jраспределено равномерно в интервале (0, 100L). Для каждого значения L 10 примеров генерируются со значениями p_j, равномерно распределенными в интервале (0, 100).

Для $n \in \{50, 100, 200, 250\}$ пять примеров генерируется для каждого $L \in \{0{,}1, 0{,}5, 0{,}8, 1, 1{,}5, 1{,}8, 2\}$. При этом значения p_j равномерно распределены в интервале (0, 100), а значения s_j– в интервале (0, 100L).

Для работы с моделями M1 и M2 использовался пакет CPLEX 10.1, при этом выделялась рабочая память 2 ГБ. Алгоритм SA реализован на языке java с использованием среды JDK 1.3.0, используемая оперативная память 2 ГБ, используемый процессор Intel(R) Core(TM) i5-2430M 2.4 GHz.

В табл. 1–4 первый столбец содержит число требований n, второй показывает значение L, третий указывает на тестируемый объект, столбцы с четвертого по шестой указывают минимальное, среднее и максимальное время счета, в столбцах с седьмого по девятый указаны минимальное, среднее и максимальное значение отношения $\sum C_j / LB$ либо $\sum C_j / \sum C_j^*$, если оптимальное значение $\sum C_j^*$ удается определить.

Таблица 1

n	*L*	mod/alg	mintime, s	averagetime, s	maxtime, s	min $\sum C_j / \sum C_j^*$	average $\sum C_j / \sum C_j^*$	max $\sum C_j / \sum C_j^*$
8	0,1	M1	33,0	48,4	59,7	1,00	1,00	1,00
		M2	0,0	0,1	0,2	1,00	1,02	1,05
		SA	9,5	10,5	13,4	1,00	1,00	1,00

Окончание табл. 1

n	*L*	mod/alg	mintime, s	averagetime, s	maxtime, s	min $\sum C_j / \sum C_j^*$	average $\sum C_j / \sum C_j^*$	max $\sum C_j / \sum C_j^*$
8	0,5	M1	2,4	20,8	58,3	1,00	1,00	1,00
		M2	0,0	0,1	0,1	1,00	1,01	1,02
		SA	10,6	13,2	15,4	1,00	1,00	1,00
	0,8	M1	1,5	9,1	18,8	1,00	1,00	1,00
		M2	0,0	0,1	0,1	1,00	1,01	1,03
		SA	11,0	13,8	15,5	1,00	1,00	1,00
	1,0	M1	2,6	8,2	18,4	1,00	1,00	1,00
		M2	0,0	0,0	0,1	1,00	1,00	1,00
		SA	13,2	14,3	15,7	1,00	1,00	1,00
	1,5	M1	0,6	2,8	6,0	1,00	1,00	1,00
		M2	0,0	0,0	0,1	1,00	1,02	1,06
		SA	13,0	15,0	15,9	1,00	1,00	1,00
	1,8	M1	0,3	2,5	6,5	1,00	1,00	1,00
		M2	0,1	1,1	2,2	1,00	1,00	1,01
		SA	15,5	15,0	11,7	1,00	1,00	1,00
	2,0	M1	0,3	2,4	6,0	1,00	1,00	1,00
		M2	0,0	0,0	0,0	1,00	1,00	1,00
		SA	12,5	14,9	16,5	1,00	1,00	1,00

Изтабл. 1 видно, чтомодельM1 иалгоритмSAнаходят оптимум достаточно быстро. Решения модели M2 часто не точны,однако этамодельработает очень быстро. Заметно, чтовремяработымоделиM1 уменьшаетсясростом*L*, аалгоритмаSA,наоборот, увеличивается.

Таблица 2

n	*L*	mod/alg	mintime,s	avetime, s	maxtime, s	min $\sum C_j / LB$	ave $\sum C_j / LB$	max $\sum C_j / LB$
20	0,1	M1	750,0	750,1	750,4	1,04	1,07	1,13
		M2	0,0	0,1	0,5	1,00	1,01	1,02
		SA	41,4	44,1	48,2	1,00	1,00	1,01
	0,5	M1	750,0	750,6	752,7	1,07	1,09	1,13
		M2	0,4	28,4	96,0	1,01	1,05	1,13
		SA	55,5	60,4	65,1	1,01	1,03	1,07
	0,8	M1	750,0	750,0	750,0	1,05	1,09	1,11
		M2	0,2	10,3	29,7	1,02	1,05	1,11
		SA	56,3	62,7	71,6	1,01	1,03	1,06
	1,0	M1	750,0	750,0	750,2	1,09	1,14	1,19
		M2	10,6	91,5	340,9	1,05	1,08	1,10
		SA	57,6	64,9	68,6	1,05	1,06	1,08
	1,5	M1	750,0	750,1	750,3	1,10	1,14	1,20
		M2	2,1	8,7	17,2	1,03	1,07	1,10

		SA	61,4	66,0	67,6	1,03	1,06	1,10
	1,8	M1	750,0	750,0	750,0	1,06	1,10	1,16
		M2	0,1	1,1	2,2	1,00	1,04	1,06
		SA	52,7	58,4	64,2	1,00	1,03	1,05
	2,0	M1	750,0	750,4	751,9	1,02	1,06	1,15
		M2	0,0	0,9	3,1	1,01	1,03	1,08
		SA	62,3	70,9	79,4	1,01	1,03	1,07

Для*n*=20 небольшой проигрыш в качествеполученных решений для моделиM2 компенсируется в большинстве случаев значительным опережением по времени.По времени M2 опережает и SA,и M1.

Для *n*=50, 100, 200 модель M1 оказалась не в состоянии за отведенное время сгенерировать допустимое решение для большинства примеров, поэтому далее сравнивается лишь эвристика SAс моделью M2.

Таблица 3

n	*L*	mod/alg	min time, s	ave time, s	max time, s	min $\sum C_j/LB$	ave $\sum C_j/LB$	max $\sum C_j/LB$
50	0,1	M2	0	77	238	1,00	1,00	1,01
		SA	369	459	586	1,00	1,00	1,00
	0,5	M2	177	980	1875	1,00	1,04	1,06
		SA	372	547	734	1,00	1,01	1,02
	0,8	M2	1069	1680	1876	1,03	1,06	1,11
		SA	463	510	619	1,02	1,03	1,04
	1	M2	1025	1622	1875	1,06	1,08	1,11
		SA	444	529	759	1,05	1,07	1,09
	1,5	M2	544	983	1531	1,03	1,05	1,10
		SA	331	436	509	1,03	1,05	1,08
	1,8	M2	577	664	782	1,03	1,05	1,07
		SA	336	410	540	1,02	1,05	1,07
	2,0	M2	531	609	691	1,02	1,06	1,11
		SA	338	420	519	1,02	1,05	1,10

Для*n*=50 алгоритмSAоказываетсялучшимиповремени, ипокачеству,однако для значения *L*=0,1 модель M2 предпочтительнее.

Таблица 4

n	*L*	mod/alg	min time, s	ave time, s	max time, s	min $\sum C_j/LB$	ave $\sum C_j/LB$	max $\sum C_j/LB$
100	0,1	M2	15	233	633	1,00	1,00	1,00
		SA	1792	1865	1960	1,00	1,00	1,00
	0,5	M2	711	1039	1400	1,01	1,02	1,04
		SA	1843	2065	2212	1,00	1,01	1,01
	0,8	M2	1718	1847	2064	1,03	1,05	1,07
		SA	1592	1764	2086	1,01	1,02	1,04
	1	M2	1702	1861	2080	1,05	1,07	1,11
		SA	1925	2041	2332	1,02	1,04	1,07
	1,5	M2	1733	1837	1944	1,03	1,06	1,09
		SA	1525	1778	1886	1,02	1,05	1,07
	1,8	M2	1310	1683	1998	1,04	1,05	1,06

		SA	1561	1779	2432	1,03	1,04	1,05
	2,0	M2	920	1123	1354	1,01	1,03	1,05
		SA	1561	1833	2349	1,01	1,03	1,05

Для n=100и L=0,8, 1, 1,5 алгоритм SA показывает лучшие результаты, чем модель M2. Однако для L=0,1, 0,5, 2,0 модель M2 предпочтительнее, чем SA.

В табл. 5 и 6 предельное время задано во втором столбце.

Таблица 5

n	time,s	L	Mod/alg	min $\sum C_j/LB$	ave $\sum C_j/LB$	max $\sum C_j/LB$
200	3600	0,1	M2	1,00	1,00	1,00
			SA	1,00	1,00	1,00
		0,5	M2	1,02	1,02	1,03
			SA	1,01	1,01	1,01
		0,8	M2	1,03	1,05	1,08
			SA	1,01	1,02	1,02
		1	M2	1,04	1,06	1,08
			SA	1,02	1,02	1,04
		1,5	M2	1,06	1,08	1,11
			SA	1,05	1,07	1,09
		1,8	M2	1,03	1,04	1,06
			SA	1,02	1,03	1,05
		2,0	M2	1,02	1,05	1,07
			SA	1,02	1,04	1,06

Дляn=200 алгоритмSAоказываетсялучше,чем модель M2, во всех случаях.

Таблица 6

n	time,s	L	mod/alg	min $\sum C_j/LB$	ave $\sum C_j/LB$	max $\sum C_j/LB$
250	3600	0,1	M2	1,00	1,00	1,00
			SA	1,00	1,00	1,00
		0,5	M2	1,01	1,04	1,06
			SA	1,01	1,01	1,02
		0,8	M2	1,04	1,06	1,08
			SA	1,01	1,02	1,02
		1	M2	1,06	1,07	1,08
			SA	1,02	1,02	1,03
		1,5	M2	1,04	1,07	1,12
			SA	1,03	1,05	1,09
		1,8	M2	1,03	1,05	1,06
			SA	1,02	1,03	1,04
		2,0	M2	1,02	1,05	1,07
			SA	1,02	1,04	1,05

Для n=250 алгоритм SAпревосходит модель M2.

Заключение

Для примеров с числом требований, не превосходящим 100, модель М2 показывает очень хорошие результаты. Учитывая предельно простую структуру строящихся расписаний, этот факт выглядит любопытным.Между тем критерий $\sum C_j$ более сложен для SAпо сравнению с критерием $C_{\max}$. Метод имитации отжига является достаточно эффективным для задачи $P2,S1||C_{\max}$ и работает с примерами, содержащими до 1000 требований [9].

Списоклитературы

1. Hall, N. Parallel machine scheduling with a common server / N. Hall, C. Potts, C. Sriskandarajah//Discrete Applied Mathematics. – 2000. – Vol.102. – P. 223–243.

2. Complexity results for parallel machine problems with a single server / P. Brucker[et al.] //Journal of Scheduling. –2002. – Vol. 5. – P. 429–457.

3. Kravchenko, S.A. A heuristic algorithm for minimizing mean flow time with unit setups / S.A. Kravchenko, F. Werner // Information Processing Letters. – 2001. – Vol. 79. – P. 291–296.

4. Wang, G. An approximation algorithm for parallel machine scheduling with a common server / G. Wang, T.C.E. Cheng // J. of the Operational Research Society. – 2001. – Vol. 52. – P. 234–237.

5.Weng, M.X. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective / M.X. Weng, J. Lu, H. Ren // International J. of Production Economics. – 2001. – Vol. 70. – P. 215–226.

6. Dunstall, S.Heuristic methods for the identical parallel machine flowtime problem with set-up times/ S. Dunstall, A. Wirth // Computers & Operations Research. –2005. – Vol. 32. – P. 2479–2491.

7. Azizoglu, M.Scheduling parallel machines to minimize weighted flowtime with family set-up times / M. Azizoglu, S. Webster // International J. of Production Research. –2003.–Vol. 41. – P. 1199–1215.

8. Guirchoun,S.Total completion time minimization in a computer system with a server and two parallel processors/ S. Guirchoun, P. Martineau, J.-C.Billaut// Computers & Operations Research. – 2005. – Vol. 32. – P. 599–611.

9. Hasani, K. Two heuristics for minimizing the makespan for the two-machine scheduling problem with a single server / K. Hasani, S.A. Kravchenko, F. Werner. – Magdeburg, 2013. – 20 p. – (Preprint / Otto-von-Guericke-University Magdeburg, Faculty of Mathematics ;№ 08/13).

Поступила 12.07.2013

[1]*Университет Магдебурга, Германия*

[2]*Объединенный институт проблем информатики НАН Беларуси, Минск, Сурганова,6*
e-mail: kravch@newman.bas-net.by

[3]*ИсламскийуниверситетАзад, Иран*

F. Werner, S.A. Kravchenko, K. Hasani

MINIMIZING MEAN FLOW TIME FOR THE TWO-MACHINE SCHEDULING PROBLEM WITH A SINGLE SERVER

In this paper, we consider the problem of scheduling a set of jobs on two parallel machines to minimize the sum of completion times. Each job requires a setup which must be done by a single server. It is known that this problem is strongly NP-hard. We proposed two mixed integer linear programming models and a simulated annealing algorithm. The performance of these algorithms is evaluated for instances with up to 250 jobs.