УДК 519.8
DOI: 10.15827/2311-6749.17.4.11

АЛГОРИТМ ВЕТВЕЙ И ГРАНИЦ ДЛЯ ЗАДАЧИ О ФОРМИРОВАНИИ ПРОИЗВОДСТВЕННЫХ ЯЧЕЕК

И.Е. Уткина, преподаватель, iutkina@hse.ru;
М.В. Бацын, к.ф-м.н., доцент, mbatsyn@hse.ru
(Национальный исследовательский университет Высшая школа экономики, ул. Печерская, 25/12б, г. Нижний Новгород, 603155, Россия)

Задача о формировании производственных ячеек является NP-трудной задачей оптимизации ячеечных производственных систем. Из-за большой вычислительной сложности представленной задачи было создано множество эвристических алгоритмов, но малое количество точных.

В статье предлагается метод ветвей и границ, который находит точное решение для текущей задачи, используя групповую эффективность в качестве целевой функции. Для линеаризации этой целевой функции используется метод Динкельбаха. Представленный алгоритм находит оптимальные решения для 24 из 35 известных тестовых данных из литературы, а для оставшихся находит хорошее решение, близкое к известному. Различие от лучшего известного решения всегда меньше 1,5 % в значении целевой функции.

Ключевые слова: *формирование производственных ячеек, бикластеризация, метод ветвей и границ, метод Динкельбаха, точное решение.*

Первая работа на тему групповых технологий в производстве была написана в 1925 году [1]. В России такое исследование было представлено только в 1933 году [2]. Один из главных вопросов при рассмотрении групповых технологий – это поиск оптимального распределения станков и деталей в производственные ячейки таким образом, чтобы максимизировать обработку внутри цехов и минимизировать передвижение деталей между цехами. Эта задача называется *задачей о формировании производственных ячеек* (ЗФПЯ) [3]. Максимизация групповой эффективности считается хорошей целевой функцией для этой задачи, так как она комбинирует обе цели [4].

В работе [5] представлен подход, основанный на анализе потока продукции, для решения ЗФПЯ, описаны групповые технологии и ЗФПЯ. Было создано множество эвристических алгоритмов для решения этой задачи [6–9] и почти ни одного точного с незаданным количеством цехов и целевой функцией, основанной на групповой эффективности. В [10] рассмотрен один из простейших вариантов ЗФПЯ – задача о распределении станков, в которой необходимо распределить станки по заданному количеству цехов, минимизируя общее расстояние Хэмминга между станками внутри цехов. Представлен точный алгоритм A* для такой формулировки ЗФПЯ, предложен метод ветвей и границ для ЗФПЯ с нефиксированным количеством цехов, с ограничением количества станков внутри цеха и целевой функцией, максимизирующей размеры так называемых взаимно разделяемых ячеек. В работах [11, 12] также представлен метод ветвей и границ для задачи распределения станков.

Один из существующих точных подходов для ЗФПЯ с бикластеризацией станков и деталей и групповой эффективностью в качестве целевой функции был описан в [13]. Авторы предложили свести задачу дробного программирования ЗФПЯ к нескольким задачам *целочисленного линейного программирования* (ЦЛП) с использованием метода Динкельбаха [14] и решить каждую такую задачу с помощью пакета CPLEX. Однако они рассматривали эту задачу с заранее заданным количеством ячеек, что является упрощением. Такая же упрощенная формулировка ЗФПЯ рассматривается в [15].

Точный алгоритм для ЗФПЯ в ее исходной формулировке с переменным числом ячеек и с групповой эффективностью в качестве целевой функции представлен в [16]. В работе [17] рассмотрена ЗФПЯ с переменным числом ячеек как задача двухкритериальной оптимизации и разработан точный алгоритм, который находит Парето-фронт решений.

В настоящей статье предлагается эффективный алгоритм ветвей и границ для ЗФПЯ с переменным числом ячеек и с целевой функцией групповой эффективности. Чтобы получить хорошую верхнюю границу, задача линеаризуется с применением подхода Динкельбаха. Для вычисления верхней границы предлагается релаксация линеаризованной задачи, которую можно эффективно решить за полиномиальное время. Удалось получить оптимальные решения для 24 из 35 тестовых задач. Начальное исследование с более простым подходом и меньшими результатами отражено в [18].

Следует отметить, что ЗФПЯ в своей классической формулировке представляет собой задачу бикластеризации, в которой станки и детали одновременно группируются в ячейки. Таким образом, предложенный подход может быть также применен к задачам бикластеризации в интеллектуальном анализе данных [19].

Описание задачи

Целью ЗФПЯ является нахождение оптимального разбиения станков и деталей на группы (производственные цеха, ячейки) для сведения к минимуму перемещения деталей из одного цеха в другой и максимизации их обработки внутри цехов. Данные для задачи представляют собой матрицу A, которая содержит нули и единицы. Размер матрицы равен $m \times p$, где m – количество станков, а p – количество деталей. Элемент a_{ij} матрицы равен единице, если деталь j должна обрабатываться на станке i. Необходимо минимизировать количество нулей внутри ячеек (недогрузка станков) и количество единиц вне ячеек (перемещение деталей между цехами).

В литературе известно несколько целевых функций, объединяющих эти две цели. Целевая функция, которая обеспечивает хорошее сочетание этих целей и широко используется в литературе, – это групповая эффективность, предложенная в работе [4]:

$$f = \frac{n_1^{in}}{n_1 + n_0^{in}} \to \max, \tag{1}$$

где n_1 – количество единиц в исходной матрице; $n_1{}^{in}$ – количество единиц внутри ячеек; $n_0{}^{in}$ – количество нулей внутри ячеек.

Представим математическую модель для ЗФПЯ (см. также [16]).

Переменные:

$$x_{ik} = \begin{cases} 1, & \text{если станок } i \text{ определен в ячейку } k, \\ 0 & \text{иначе,} \end{cases} \tag{2}$$

$$y_{jk} = \begin{cases} 1, & \text{если деталь } j \text{ определена в ячейку } k, \\ 0 & \text{иначе.} \end{cases} \tag{3}$$

Целевая функция:

$$\max \frac{n_1^{in}}{n_1 + n_0^{in}} \tag{4}$$

Ограничения:

$$n_1^{in} = \sum_{k=1}^{c}\sum_{i=1}^{m}\sum_{j=1}^{p} a_{ij} x_{ik} y_{jk}; \tag{5}$$

$$n_0^{in} = \sum_{k=1}^{c}\sum_{i=1}^{m}\sum_{j=1}^{p} (1 - a_{ij}) x_{ik} y_{jk}; \tag{6}$$

$$\sum_{k=1}^{c} x_{ik} = 1 \quad \forall i = 1, ..., m; \tag{7}$$

$$\sum_{k=1}^{c} y_{jk} = 1 \quad \forall j = 1, \ldots, p; \tag{8}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{p} x_{ik} y_{jk} \geq \sum_{i=1}^{m} x_{ik}, \quad \forall k = 1, \ldots, c; \tag{9}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{p} x_{ik} y_{jk} \geq \sum_{j=1}^{p} y_{jk}, \quad \forall k = 1, \ldots, c. \tag{10}$$

Здесь $c = \min(m, p)$ – максимально возможное количество ячеек. Ограничения (7) и (8) требуют, чтобы каждый станок и каждая деталь относились ровно к одной ячейке. Ограничения (9) и (10) требуют, чтобы не было ячеек, которые содержат только станки или только детали.

Для линеаризации целевой функции используем метод Динкельбаха и рассмотрим следующую целевую функцию: $\tilde{f} = n_1^{in} - \lambda\left(n_1 + n_0^{in}\right) \to \max$, где λ на каждой итерации равно текущему лучшему решению для исходной ЗФПЯ. На первой итерации λ равно значению групповой эффективности в тривиальном решении, в котором все станки и детали назначены в одну большую ячейку:

$$\lambda = \frac{n_1}{n_1 + n_0}.$$

Чтобы найти оптимальное решение $\tilde{f}^*$ линеаризованной задачи, применим метод ветвей и границ.

Если оптимальное решение $\tilde{f}^*$ равно нулю, то для любого возможного решения $\tilde{f} \leq \tilde{f}^* = 0,$ сдедовательно,

$$n_1^{in} - \lambda\left(n_1 + n_0^{in}\right) \leq 0 \quad \Leftrightarrow \quad f = \frac{n_1^{in}}{n_1 + n_0^{in}} \leq \lambda.$$

Это означает, что λ является оптимальным решением для ЗФПЯ, так как любое допустимое решение $f \le \lambda$. В этом случае останавливаем алгоритм.

Если $\tilde{f}^* > 0$, тогда

$$n_1^{in} - \lambda\left(n_1 + n_0^{in}\right) > 0 \Leftrightarrow f = \frac{n_1^{in}}{n_1 + n_0^{in}} > \lambda .$$

Это свидетельствует о том, что найденное решение f лучше текущего лучшего решения λ. Заметим, что нет необходимости находить оптимальное $\tilde{f}^*$, рассматривая все ветки в методе ветвей и границ – любое допустимое решение $\tilde{f} > 0$ может быть использовано. В этом случае устанавливаем $\lambda = f$ и повторно запускаем решение линеаризованной задачи методом ветвей и границ с новым значением λ.

Заметим, что $\tilde{f}^*$ не может быть отрицательным, так как

$$\tilde{f} \le \tilde{f}^* < 0 \quad \Rightarrow \quad \frac{n_1^{in}}{n_1 + n_0^{in}} < \lambda,$$

но уже есть допустимое решение $f = \lambda$.

Метод ветвей и границ

Псевдокод алгоритма приведен далее в алгоритмах 1–4. Для решения линеаризованной задачи $\tilde{f} = n_1^{in} - \lambda\left(n_1 + n_0^{in}\right) \to \max$ предлагается метод ветвей и границ. Ветвление происходит по станкам, а граница вычисляется как решение полиномиальной релаксации задачи. Для ускорения алгоритма не следует искать оптимальное решение $\tilde{f}^*$ на каждом шаге метода Динкельбаха (см. алгоритм 1), а как только найдено допустимое решение $\tilde{f} > 0$ (строка 7 алгоритма 2), надо остановиться.

Для описания ветвей используем вектор M ($1 \times m$), который содержит распределение станков по ячейкам. Элемент M_i определяет, в какую ячейку назначен станок i. $M_i \in \{1, \ldots, c\}$, где $c = \min(m, p)$. Например, $M = (1, 2, 3, 1, 0, \ldots, 0)$ означает, что станки 1 и 4 назначены в ячейку 1, станок 2 – в ячейку 2, станок 3 – в ячейку 3, а другие станки еще не распределены.

Алгоритм 1 использует подход Динкельбаха. Текущее лучшее распределение станков сохраняется в векторе M^*, значение целевой функции – в переменной f^*, а количество единиц и нулей внутри ячеек – в переменных n_1^{in*} и n_0^{in*}. Текущее лучшее решение линеаризованной задачи сохраняется в переменной $\tilde{f}^*$. Так как числа с плавающей точкой имеют ограниченную точность, сравниваем $\tilde{f}^*$ с величиной 0,00001 вместо нуля. Это значение достаточно мало, чтобы гарантировать $\tilde{f}^* = 0$, так как

$$\tilde{f} = n_1^{in} - \frac{n_1^{in*}}{n_1 + n_0^{in*}}\left(n_1 + n_0^{in}\right) = \frac{1}{n_1 + n_0^{in}}\left(n_1^{in}\left(n_1 + n_0^{in*}\right) - n_1^{in*}\left(n_1 + n_0^{in}\right)\right)$$

и все значения в скобках целые. Поэтому $\tilde{f}$ будет положительным, только если

$$n_1^{in}\left(n_1 + n_0^{in*}\right) - n_1^{in*}\left(n_1 + n_0^{in}\right) \ge 1 \Rightarrow \tilde{f} \ge \frac{1}{n_1 + n_0^{in}} \ge \frac{1}{n_1 + n_0} = \frac{1}{mp}$$

Таким образом, для любого решения, включая текущее лучшее $\tilde{f}^*$, значение будет положительным, только если оно не меньше $1/mp$.

В алгоритме 1 начальное решение исходной задачи выбирается равным $f^* = \lambda = n_1/(n_1 + n_0)$, что означает распределение всех станков и деталей в одну ячейку. Далее для каждой итерации, пока $\tilde{f}^*$ не ноль, решаем линеаризованную задачу с новым значением λ, используя метод ветвей и границ. Когда $\tilde{f}^*$ становится нулем, останавливаем алгоритм, так как это означает, что текущее лучшее решение $f^* = \lambda$ является оптимальным для исходной задачи.

Алгоритм 1. Основной алгоритм, основанный на методе Динкельбаха:

function DinkelbachProcedure()

$\tilde{f}^* \leftarrow 1$ — Текущее лучшее решение для линеаризованной задачи

$c = \leftarrow \min(m, p)$

$n_1^{in*} \leftarrow n_1, n_0^{in*} \leftarrow n_0$

$M^* \leftarrow (1, \ldots, 1)$ — Начинаем с тривиального решения с одной ячейкой

while ($\tilde{f}^* > 0.00001$) **do** — Используем 0.00001 из-за ограниченной точности

$M \leftarrow (1, 0, \ldots, 0)$ Назначить 1-й станок в ячейку 1
$k \leftarrow 2$ Количество ячеек для следующего ветвления
$i \leftarrow 2$ Номер станка для следующего ветвления
$\lambda \leftarrow n_1^{in*} / \left(n_1 + n_0^{in*}\right)$
$\tilde{f}^* \leftarrow -\infty$
$f^* \leftarrow \lambda$ Текущее лучшее решение исходной задачи
BRANCH-AND-BOUND(M, k, i)
return M^*, f^*

Ветвление. Пусть k – количество ячеек в текущем частичном решении. Алгоритм начинает с распределения первого станка в ячейку 1. Потом он использует следующий нераспределенный станок и определяет его в существующие ячейки 1, ..., k или создает новую (k+1) для этого станка. Например, в вершине, когда M = (1, 2, 3, 1, 0, …,0), получим 4 ветви: (1, 2, 3, 1, 1, 0, …,0), (1, 2, 3, 1, 2, 0, …,0), (1, 2, 3, 1, 3, 0, …,0) и (1, 2, 3, 1, 4, 0, …,0) (строка 2 алгоритма 2).

Листья дерева поиска содержат полные решения, а остальные вершины – частичные. Полное дерево поиска без использования границ зависит только от количества станков. Заметим, что ячейка k + 2 в частичном решении никогда не используется, если ячейка k + 1 еще не была использована в текущем решении (последняя строка алгоритма 2). Это гарантирует, что не будут рассматриваться одинаковые решения, которые различаются только нумерацией ячеек.

Следуем сильной стратегии ветвления: перед тем как выбрать ветвь для следующего шага, подсчитаем верхнюю границу каждой ветви и перейдем в ту, которая дает максимальное значение (строка 4 алгоритма 2).

Алгоритм 2. Метод ветвей и границ:

function BRANCH-AND-BOUND(M, k, i)
for (l = 1, ..., k) **do**
$M_i \leftarrow l$ Определить станок i в ячейку l
$\{UB, n_1^{in}, n_0^{in}, C\}_l \leftarrow$ upperBound$\left(M, \max(l, k-1), \varnothing\right)$
$L \leftarrow$sort(UB) L хранит отсортированные индексы L_1=argmax$_l UB_l$
for (u = 1, ..., k) **do**
if $\left(\tilde{f}^* \geq 0\right)$ **then** Найдено решение лучше, чем λ, или λ оптимальное
return Останавливаем метод, переходим к следующей итерации метода Динкельбаха
$l \leftarrow L_u$
if $\left(UB_l \leq \tilde{f}^*\right)$ **then**
return Эта и все следующие верхние границы $\leq \tilde{f}^*$, поэтому отбрасываем эти ветви
$M_i \leftarrow l$ Определяем станок i в ячейку l
$\{UB, n_1^{in}, n_0^{in}, C\}_l \leftarrow$ upperBound(M, max((l, k – 1),C$_l$)
if $\left(UB_l \leq \tilde{f}^*\right)$ **then**
continue Отсекаем ветвь после подсчета верхней границы с решением задачи о назначениях
if (i = m) **then** Построено полное решение с $\tilde{f} = UB > \tilde{f}^*$
$\tilde{f}^*, n_1^{in*}, n_0^{in*} \leftarrow \{UB, n_1^{in}, n_0^{in}\}_l$
continue
BRANCH-AND-BOUND(M, min((k + 1, c), i + 1) Переходим к следующему станку

Верхняя граница. Чтобы получить оценку сверху для данного частичного решения, предлагается релаксация линеаризованной задачи ФПЯ, которая может быть решена с помощью полиномиального алгоритма. Релаксированная задача формулируется следующим образом. Дано частичное решение, в котором только некоторые из станков уже назначены в ячейки. Например, на рисунке 1 станки 1 и 2 назначаются в ячейку 1, а станок 3 – в ячейку 2. Необходимо назначить все детали в существующие ячейки или в новую ячейку. В релаксированной задаче допускаем независимое назначение деталей: например, деталь 1 может быть назначена в ячейку 1 со станками 1, 2 (уже назначенными в ячейку 1), а также со станком 4, а деталь 8 может быть назначена в ту же самую ячейку 1 с другими станками – 1, 2 и 5.

В примере из рисунка 1 деталь 1 может быть назначена в ячейку 1 с двумя единицами от станков 1 и 2 и третьей единицей от станка 4, который потенциально может принадлежать и ячейке 1. Поэтому в лучшем случае назначение детали 1 в ячейку 1 даст 3 единицы и 0 нулей внутри ячеек. Обозначим этот вариант как (3, 0). Также назначение детали 1 в ячейку 2 дает в лучшем случае 1 единицу (от станка 4) и 1 ноль (от станка 3) внутри ячеек. Назначение в новую ячейку дает 1 единицу (от станка 4) и 0 нулей внутри ячеек. Итак, есть 3 альтернативы (3,0), (1,1) и (1,0) для детали 1.

Для выбора между двумя альтернативами надо посчитать их вклад в целевую функцию $\tilde{f} = n_1^{in} - \lambda\left(n_1 + n_0^{in}\right) = \left(n_1^{in} - \lambda n_0^{in}\right) - \lambda n_1$. Так как λn_1 – константа, очевидно, что вклад альтернативы (a, b) (где $n_1^{in} = a, n_0^{in} = b$), равен $a - \lambda b$. Пусть λ = 0.5 в нашем примере. Тогда для детали с альтернативами (2,1), (1,1), и (1,0) получаем вклады 1.5, 0.5, и 1.0 соответственно. Значит, альтернатива (2,1) наилучшая (см. алгоритм 3).

Оптимальное решение для релаксированной задачи показано на рисунке 2. Это решение недопустимо для исходной ЗФПЯ, потому что из-за независимого распределения получены непрямоугольные ячейки, пересекающиеся друг с другом. Так как это оптимальное решение релаксированной задачи, оно дает верхнюю границу для исходной задачи: UB = 18.5 – 0.5·20 = 8.5.

	1	2	3	4	5	6	7	8
1	1	1	1	1	1	0	0	1
2	1	1	0	1	1	0	0	1
3	0	0	0	0	1	0	0	0
4	1	0	1	1	1	1	1	0
5	0	0	0	0	0	0	1	1

Рис. 1. Частичное решение для ЗФПЯ

	1	2	3	4	5	6	7	8
1	1	1	1	1	1	0	0	1
2	1	1	0	1	1	0	0	1
3	0	0	0	0	1	0	0	0
4	1	0	1	1	1	1	1	0
5	0	0	0	0	0	0	1	1

Рис. 2. Оптимальное решение релаксированной ЗФПЯ

Алгоритм 3. Подсчет верхней границы:

function upperBound(M, k, C)
if (C≠∅) **then**
[j_l] ← solveAP(C) — Запускаем этот метод без решения ЗН и заполняем C
$n_1^{in} \leftarrow 0, n_0^{in} \leftarrow 0,\ UB \leftarrow -\lambda n_1$
for (j = 1, ..., p) **do**
if ($j \in [j_l]$) **then**
$l \leftarrow i | j_i = j$ — j_l индекс детали, определенной в ячейку l
$a^*, b^* \leftarrow$ countOnesAndZeroes(M, l) — получить ячейку, в которую определена деталь j
$\max \leftarrow a^* - \lambda b^*$
else
max ← 0, $a^* \leftarrow 0$, $b^* \leftarrow 0$
for (l = 1, ..., k) **do**
$a, b \leftarrow$ countOnesAndZeroes(M, l)
$C_{lj} \leftarrow \lambda b - a$
if ($a - \lambda b >$ max) **then**
max ← $a - \lambda b$, $a^* \leftarrow a$, $b^* \leftarrow b$
$UB \leftarrow UB +$ max
$n_1^{in} \leftarrow n_1^{in} + a^*$
$n_0^{in} \leftarrow n_0^{in} + b^*$
return $UB, n_1^{in}, n_0^{in}, C$

Алгоритм 4. Подсчет единиц и нулей внутри ячеек для релаксированной задачи:

function countOnesAndZeroes(M, l)
$a \leftarrow 0, b \leftarrow 0$
for (i = 1, ..., m) **do**
if ($M_i = l$) **then**

```
        if (A_ij = 1) then
          a ← a + 1
        else
          b ← b + 1
      if (M_i = 0) then
        if (A_ij = 1) then
          a ← a + 1
  return a, b
```

Далее находим наилучшее назначение всех деталей в ячейки и вычисляем целевую функцию как сумму вкладов деталей минус λn_1, что дает нам верхнюю границу. Однако после такого назначения могут получиться ячейки со станками, но без деталей. Такие ячейки не допускаются в классической формулировке ЗФПЯ. Каждая ячейка, имеющая станки, должна иметь хотя бы одну деталь. Целевая функция линеаризованной задачи линейна, а назначение одной детали вносит вклад, не зависящий от другой детали. Это означает, что мы можем сначала выбрать оптимально одну деталь для каждой ячейки, максимизируя общий вклад в целевую функцию этих выбранных деталей, а затем назначить каждую оставшуюся деталь в лучшую для нее ячейку без каких-либо ограничений. Очевидно, что задача выбора одной детали для каждой ячейки эквивалентна линейной задаче о назначениях (ЗН), в которой стоимость назначения равна вкладу детали в целевую функцию со знаком «минус», то есть $\lambda b - a$ (см. алгоритм 3). Чтобы сбалансировать матрицу затрат (сделать ее квадратной) в этой задаче, добавляем фиктивные ячейки, для которых все стоимости назначения равны $-\infty$.

Для решения задачи о назначениях применяем алгоритм Йонкера–Волгенанта [20]. Перед запуском этого алгоритма проверяем решение релаксированной задачи, в котором могут быть ячейки без деталей. Если таких ячеек нет или эта верхняя граница уже позволяет отбросить текущую ветвь, не решаем задачу о назначениях, чтобы уменьшить время вычислений.

В примере на рисунке 2 ячейка 2 содержит станок 3, но не имеет деталей. Поэтому нужно назначить одну деталь каждой ячейке, решая задачу о назначениях. Матрица затрат для нашего примера представлена на рисунке 3 (ячейки 4, 5, 6, 7, 8 добавляются для балансировки матрицы).

Ячейки \ Детали	1	2	3	4	5	6	7	8
1	-3	-2	-1.5	-3	-3	0	-1	-3
2	-0.5	0.5	-0.5	-0.5	-2	-0.5	-1.5	-0.5
3	-1	0	-1	-1	-1	-1	-2	-1
4								
5								
6				$-\infty$				
7								
8								

Рис. 3. Матрица затрат

Матрица стоимостей для задачи о назначениях

Одним из оптимальных решений этой задачи о назначениях является назначение детали 1 в ячейку 1, детали 5 в ячейку 2 и детали 7 в ячейку 3 с вкладом в верхнюю границу, равным 3 + 2 + 2 = 7. Каждая другая деталь назначается без каких-либо ограничений в ячейку, где она дает максимальный вклад в верхнюю границу. Таким образом, получаем $UB = 7 + (2 + 1.5 + 3 + 1 + 3)-0.5*20=7.5$.

Заметим, что, когда все станки распределены по ячейкам, решение релаксированной задачи с применением задачи о назначениях дает допустимое решение для исходной задачи. Поэтому получается оптимальное решение $\tilde{f}^* = UB$.

Результаты

Представленный алгоритм ветвей и границ находит точное решение для 24 из 35 известных тестовых примеров из литературы, а для остальных примеров – решение, близкое к лучшему известному решению. Результаты представлены в таблице. Все вычисления были произведены на Intel Core i7 с 16Gb RAM. Как видим, разработанный алгоритм эффективнее подходов, предложенных в работах [15, 16], даже если не учитывать, что в работе [16] в качестве начального решения используется лучшее известное решение,

а алгоритм Бруско [15] решает тестовые задачи только для нескольких фиксированных значений числа ячеек. Для результатов Бруско [15] в таблице представлено общее время, затрачиваемое на решение каждого примера для всех рассматриваемых значений числа ячеек и получение начального решения эвристикой.

На основании этих результатов можно сказать, что описываемый алгоритм работает быстрее для задач с высоким оптимальным значением целевой функции. Благодаря эффективной стратегии ветвления алгоритм быстро находит хорошие решения, и, если оптимальное значение целевой функции велико, то многие ветви отбрасываются, потому что предложенная верхняя граница обычно намного более точная для таких случаев. Например, задачи 20 и 21 имеют одинаковый размер входной матрицы, но задача 20 с оптимальным решением 0,7791 решается в течение 2 секунд, в то время как задача 21 с точным решением 0,5798 требует 207 секунд.

Сравнительный анализ результатов работы разных алгоритмов и решений

Номер	Источник	Размер входной матрицы	Лучшее известное решение	f^*	Разработанный алгоритм, время, сек.	Алгоритм Бычкова и др. (2014) [10], время, сек.	Алгоритм Бруско (2015) [6], время, сек. (ячейки)
1	King & Nakornchai (1982) [21]	5×7	0.8235	0.8235	0.00	0.63	2.87 (2–4)
2	Waghodekar & Sahu (1984) [22]	5×7	0.6957	0.6957	0.00	2.29	3.62 (2–4)
3	Seifoddini (1989) [23]	5×18	0.7959	0.7959	0.00	5.69	8.45 (2–4)
4	Kusiak (1987) [24]	6×8	0.7692	0.7692	0.00	1.86	4.11 (2–4)
5	Kusiak & Chow (1987) [25]	7×11	0.6087	0.6087	0.00	9.14	13.37 (2–6)
6	Boctor (1991) [26]	7×11	0.7083	0.7083	0.00	5.15	10.13 (2–5)
7	Seifoddini & Wolfe (1986) [27]	8×12	0.6944	0.6944	0.00	13.37	15.62 (2–5)
8	Chandrasekharan & Rajagopalan (1986a) [28]	8×20	0.8525	0.8525	0.00	18.33	11.41 (2–4)
9	Chandrasekharan & Rajagopalan (1986b) [29]	8×20	0.5872	0.5872	0.00	208.36	51.00 (2–4)
10	Mosier & Taube (1985a) [30]	10×10	0.7500	0.7500	0.00	6.25	17.83 (2–6)
11	Chan & Milner (1982) [31]	15×10	0.9200	0.9200	0.00	2.93	9.28 (2–4)
12	Askin & Subramanian (1987) [32]	14×24	0.7206	0.7206	0.04	259.19	694.76 (6–8)
13	Stanfel (1985) [33]	14×24	0.7183	0.7183	0.04	259.19	2690.22 (6–8)
14	McCormick et al. (1972) [34]	16×24	0.5326	0.5326	71.63	[б] 20829.38	24728.82 (8)
15	Srinivasan et al. (1990) [35]	16×30	[c] 0.6899	0.6899	0.27	[б] 13719.99	1305.40 (5–7)
16	King (1980) [36]	16×43	0.5753	0.5753	3.14	[б] 24930.93	-
17	Carrie (1973) [37]	18×24	0.5773	0.5773	4.65	[б] 13250.01	20854.09 (9)
18	Mosier & Taube (1985b) [38]	20×20	0.4345	[a] 0.4345	43200.00	[б] 43531.77	-

19	Kumar et al. (1986) [39]	23×20	0.5081	0.5081	1689.45	[б] 33020.13	1375621.31 (7)
20	Carrie (1973) [37]	20×35	0.7791	0.7791	1.24	[б] 11626.98	4867.04 (5–7)
21	Boe & Cheng (1991) [40]	20×35	0.5798	0.5798	207.47	[б] 33322.08	-
22	Chandrasekharan & Rajagopalan (1989) [41]	24×40	1.0000	1.0000	0.00	0.00	15.15 (7)
23	Chandrasekharan & Rajagopalan (1989) [41]	24×40	0.8511	0.8511	0.08	[б] 6916.24	64.52 (7)
24	Chandrasekharan & Rajagopalan (1989) [41]	24×40	0.7351	0.7351	21.81	[б] 14408.88	208178.22 (7)
25	Chandrasekharan & Rajagopalan (1989) [41]	24×40	0.5329	[a] 0.5329	43200.00	[б] 34524.47	-
26	Chandrasekharan & Rajagopalan (1989) [41]	24×40	0.4895	[a] 0.4861	43200.00	[б] 41140.94	-
27	Chandrasekharan & Rajagopalan (1989) [41]	24×40	0.4726	[a] 0.4600	43200.00	[б] 44126.76	-
28	McCormick et al. (1972) [34]	27×27	0.5482	[a] 0.5423	43200.00	[б] 22627.28	-
29	Carrie (1973) [37]	28×46	0.4706	[a] 0.4634	43200.00	[б] 71671.08	-
30	Kumar & Vannelli (1987) [42]	30×41	0.6331	[a] 0.6331	43200.00	[б] 22594.20	-
31	Stanfel (1985) [33]	30×50	[c] 0.5977	[a] 0.5977	43200.00	[б] 31080.82	-
32	Stanfel (1985) [33]	30×50	0.5083	[a] 0.5000	43200.00	[б] 48977.01	-
33	King & Nakornchai (1982) [21]	30×90	0.4775	[a] 0.4759	43200.00	[б] 99435.64	-
34	McCormick et al. (1972) [34]	37×53	0.6064	[a] 0.5860	43200.00	[б] 47744.04	-
35	Chandrasekharan & Rajagopalan (1987) [43]	40×100	0.8403	0.8403	305.11	[б] 24167.76	-

Примечание:

[a] Разработанный алгоритм не нашел оптимальное решение за 12 часов (43 200 сек.).

[б] Задача не решена оптимально алгоритмом Бычкова и др. [16]. В его алгоритме задача делится на набор подзадач целочисленного программирования и устанавливается ограничение 300 секунд для каждой подзадачи.

[c] В некоторых статьях про эвристики предложено решение с большим значением целевой функции, скорее всего, из-за некорректной входной матрицы.

Заключение

В статье рассматривается классическая формулировка задачи о формировании производственных ячеек, которая имеет высокую вычислительную сложность из-за следующих факторов:

– целевая функция групповой эффективности – это дробная функция, которая делает ее задачей целочисленного дробного программирования;

– это задача бикластеризации, в которой необходимо группировать одновременно станки и детали в производственные ячейки;

– оптимальное количество ячеек заранее неизвестно.

Из-за этих трудностей в большинстве работ, посвященных ЗФПЯ с переменным числом ячеек и целевой функцией групповой эффективности, представлены эвристические алгоритмы, а точные подходы в литературе практически отсутствуют. В данном точном алгоритме применяется процедура Динкельбаха и рассматривается линеаризованная задача для преодоления первой трудности. Чтобы избежать второй трудности в разработанном алгоритме ветвей и границ, сначала назначаем только станки в ячейки, а затем получаем верхнюю границу с помощью решения релаксированной полиномиальной задачи. В листовых узлах дерева поиска, когда все станки назначены в ячейки, релаксированная задача становится эквивалентной исходной линеаризованной задаче, и получается оптимальное решение с помощью полиномиального алгоритма. Третья трудность заставляет рассматривать гораздо больше ветвей, чем это было бы при фиксированном числе ячеек. Если в первом точном подходе [18] удалось точно решить только 14 тестовых задач, то новый алгоритм позволяет решить еще 10 задач больших размеров.

Исследование выполнено в лаборатории ЛАТАС, НИУ ВШЭ и поддержано грантом РНФ 14-41-00039.

Литература

1. Flanders R.E. Design manufacture and production control of a standard machine. Transactions of ASME, 1925, vol. 46, pp. 691–738.
2. Митрофанов С.П. Научные основы групповых технологий. Л.: Лениздат, 1933. 435 с.
3. Goldengorin B., Krushinsky D., Pardalos P.M. Cell formation in industrial engineering. Theory, algorithms and experiments. Springer, Optimization and Its Applications, 2013, vol. 79, 206 p.
4. Kumar K.R., Chandrasekharan M.P. Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. Intern. Jour. Production Research, 1990, vol. 28, no. 2, pp. 233–243.
5. Burbidge J.L. The new approach to production. Prod. Eng., 1961, pp. 3–19.
6. Goncalves J.F., Resende M.G.C. An evolutionary algorithm for manufacturing cell formation. Computers & Industrial Engineering, 2004, vol. 47, pp. 247–273.
7. James T.L., Brown E.C., Keeling K.B. A hybrid grouping genetic algorithm for the cell formation problem. Computers & Operations Research, 2007, vol. 34, no. 7, pp. 2059–2079.
8. Bychkov I., Batsyn M., Sukhov P., Pardalos P.M. Heuristic algorithm for the cell formation problem. In: Models, Algorithms, and Technologies for Network Analysis. Springer Proc. Mathematics & Statistics, 2013, vol. 59, pp. 43–69.
9. Paydar M.M., Saidi-Mehrabad M. A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. Computers & Operations Research, 2013, vol. 40, no. 4, pp. 980–990.
10. Kusiak A., Boe J.W., Cheng C. Designing cellular manufacturing systems: branch-and-bound and A* approaches. IIE Transactions, 1993, vol. 25, no. 4, pp. 46–56.
11. Spiliopoulos K., Sofianopoulou S. An optimal tree search method for the manufacturing systems cell formation problem. Europ. Jour. Operational Research, 1998, vol. 105, pp. 537–551.
12. Arkat J., Abdollahzadeh H., Ghahve H. A new branch-and-bound algorithm for cell formation problem. Applied Mathematical Modelling, 2012, vol. 36, pp. 5091–5100.
13. Elbenani B., Ferland J.A. Cell Formation Problem Solved Exactly with the Dinkelbach Algorithm. Montreal, Quebec, 2012, pp. 1–14.
14. Dinklebach W. On nonlinear fractional programming. Management Science, 1967, vol. 13, pp. 492–498.
15. Brusco J.M. An exact algorithm for maximizing grouping efficacy in part-machine clustering. IIE Transactions, 2015, vol. 47, no. 6, pp. 653–671.
16. Bychkov I., Batsyn M., Pardalos P. Exact model for the cell formation problem. Optimization Letters, 2014, vol. 8, no. 8, pp. 2203–2210.
17. Zilinskas J., Goldengorin B., Pardalos P.M. Pareto-optimal front of cell formation problem in group technology. Jour. Global Optimization, 2015, vol. 61, no. 1, pp. 91–108.
18. Utkina I., Batsyn M., Batsyna E. A branch and bound algorithm for a fractional 0-1 programming problem. Lecture Notes in Computer Science, 2016, vol. 9869, pp. 244–255.
19. Busygin S., Prokopyev O., Pardalos P.M. Biclustering in data mining. Computers & Operations Research 2008, vol. 35, no. 9, pp. 2964–2987.
20. Jonker R., Volgenant A.A. Shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing, 1987, vol. 38, pp. 325–340.
21. King J.R., Nakornchai V. Machine-component group formation in group technology: Review and extension. Intern. Jour. Production Research, 1982, vol. 20, no. 2, pp. 117–133.
22. Waghodekar P.H., Sahu S. Machine-component cell formation in group technology MACE. Intern. Jour.

Production Research, 1984, vol. 22, pp. 937–948.

23. Seifoddini H. A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. Intern. Jour. Production Research, 1989, vol. 27 no. 7, pp. 1161–1165.

24. Kusiak A. The generalized group technology concept. International Journal of Production Research, 1987, vol. 25, no. 4, pp. 561–569.

25. Kusiak A., Chow W.S. Efficient solving of the group technology problem. Jour. of Manufacturing Systems, 1987, vol. 6, no. 2, pp. 117–124.

26. Boctor F.F. A linear formulation of the machine-part cell formation problem. Intern. Jour. Production Research, 1991, vol. 29, no. 2, pp. 343–356.

27. Seifoddini H., Wolfe P.M. Application of the similarity coefficient method in group technology. IIE Transactions, 1986, vol. 18, no. 3, pp. 271–277.

28. Chandrasekharan M.P., Rajagopalan R. MODROC: An extension of rank order clustering for group technology. Intern. Jour. Production Research, 1986, vol. 24, no. 5, pp. 1221–1233.

29. Chandrasekharan M.P., Rajagopalan R. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. Intern. Jour. Production Research, 1986, vol. 24, no. 2, pp. 451–464.

30. Mosier C.T., Taube L. The facets of group technology and their impact on implementation. OMEGA, 1985, vol. 13, no. 6, pp. 381–391.

31. Chan H.M., Milner D.A. Direct clustering algorithm for group formation in cellular manufacture. Jour. Manufacturing Systems, 1982, vol. 1, no. 1, pp. 64–76.

32. Askin R.G., Subramanian S.P. A cost-based heuristic for group technology configuration. Inter. Jour. Production Research, 1987, vol. 25, no. 1, pp. 101–113.

33. Stanfel L. Machine clustering for economic production. Engineering Costs and Production Economics, 1985, vol. 9, pp. 73–81.

34. McCormick W.T., Schweitzer P.J., White T.W. Problem decomposition and data reorganization by a clustering technique. Operations Research 1972, vol. 20, no. 5, pp. 993–1009.

35. Srinivasan G., Narendran T.T., Mahadevan B. An assignment model for the part-families problem in group technology. Intern. Jour. Production Research, 1990, vol. 28, no. 1, pp. 145–152.

36. King J.R. Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. Intern. Jour. Production Research, 1980, vol. 18, no. 2, pp. 213–232.

37. Carrie S. Numerical taxonomy applied to group technology and plant layout. Intern. Jour. Production Research, 1973, vol. 11, pp. 399–416.

38. Mosier C.T., Taube L. Weighted similarity measure heuristics for the group technology machine clustering problem. OMEGA, 1985, vol. 13, no. 6, pp. 577–583.

39. Kumar K.R., Kusiak A., Vannelli A. Grouping of parts and components in flexible manufacturing systems. Europ. Jour. Operations Research, 1986, vol. 24, pp. 387–397.

40. Boe W., Cheng C.H. A close neighbor algorithm for designing cellular manufacturing systems. International Jour. Production Research, 1991, vol. 29, no. 10, pp. 2097–2116.

41. Chandrasekharan M.P., Rajagopalan R. (). Groupability: Analysis of the properties of binary data matrices for group technology. Intern. Jour. Production Research, 1989, vol. 27, no. 6, pp. 1035–1052.

42. Kumar K.R., Vannelli A. Strategic subcontracting for efficient disaggregated manufacturing. International Jour. Production Research, 1987, vol. 25, no. 12, pp. 1715–1728.

43. Chandrasekharan M.P., Rajagopalan R. ZODIAC: An algorithm for concurrent formation of part families and machine cells. Intern. Jour. Production Research, 1987, vol. 25, no. 6, pp. 835–850.