УДК 519.6

Л.А. Казаковцев, А.Н. Антамошкин

АЛГОРИТМ ДЛЯ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ

Задача расчета оптимальной загрузки производственных линий с непрерывным производственым циклом (например, литейно-прокатное производство), согласно заданному плану выпуска продукции, представлена в виде задачи размещения на сети. Для решения задачи применен генетический алгоритм с жадной эвристикой на основе аналогичного алгоритма для p-медианной задачи.

Ключевые слова: *генетический алгоритм, дискретная задача размещения, оперативное календарное планирование, p-медианная задача.*

L.A. Kazakovtsev, A.N. Antamoshkin

ALGORITHM FOR CAPACITY SCHEDULING

A problem of optimal capacity planning of the production lines with a continuous manufacturing cycle (foundry for example) in accordance with a given product launch plan is considered as a discrete location problem on a network. For solving the problem, we use genetic algorithm with greedy heuristics based on an analogous algorithm for the p-median problem.

Key words: *genetic algorithm, discrete location problem, operations scheduling, p-median problem.*

Введение. Одной из основ эффективной работы предприятия является процедура календарного планирования производства, включая расчет производственного расписания [1]. Рассмотрим следующую задачу [2, 3]. Пусть имеется K производственных линий для выпуска L видов продукции. Производительность всех производственных линий одинакова, для l-го вида продукции линия может произвести V_l единиц продукции за смену при трех сменах в сутки. Требуется построить график с указанием вида продукции для производственной линии. Вводится матрица Z булевых констант $z_{k,l}, k = \overline{1,K}, l = \overline{1,L}$, равных 1, если k-я линия может производить l-й вид продукции, и 0 – в противном случае. Для каждого вида продукции установлен производственный план в объеме W_l единиц ($l = \overline{1,L}$), который должен быть выполнен за T_l суток. Кроме того, установлена минимальная суммарная загруженность производственного комплекса в сутки в объеме W_{min} единиц продукции. При смене вида продукции могут требоваться некоторые технологические операции продолжительностью в одну смену, в ходе которых выпуск продукции линией невозможен. Возможность безостановочной смены продукции с вида l на вид r описывается симметричной матрицей булевых констант $C_{l,r}$ размерности LxL: значение $C_{l,r} = 1$ означает необходимость останова производственной линии при смене продукции с l-го вида на r-й вид, $C_{l,r} = 0$ – возможность безостановочного переключения производства. График на I суток требуется составить так, чтобы при условии выполнения плана выпуска по видам продукции и срокам, с учетом требования минимальной загруженности, требовалось минимальное число изменений видов продукции.

В *p*-медианной задаче на сети [4, 5] требуется найти *p* узлов сети, таких, чтобы сумма взвешенных расстояний от каждого из узлов сети до ближайшего из выбранных узлов была минимальной (каждому из ребер поставлено в соответствие число – его длина). В общем случае задача NP – трудная. Генетический алгоритм с жадной эвристикой [5, 6] – эффективное средство ее решения. В настоящей работе задача составления графика загрузки производственных мощностей формулируется как задача размещения на сети. Вычислительные эксперименты показывают высокую эффективность такого подхода по сравнению с существующими методами [2].

1. Математическая постановка задачи

Приведем постановку p-медианной задачи [4]. На некоторой сети (связном графе) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, где $\mathcal{V}$ – множество узлов (вершин), а $\mathcal{E}$ – множество попарно соединяющих их ребер $E_{i,j}, i, j \in \mathcal{V}$, каждому из которых поставлено в соответствие число – его длина $\mathcal{L}_{i,j}$, определена метрика расстояния $D(i,j)$ между любой парой узлов i и j как минимальная длина пути между этими узлами (длина пути – сумма длин входящих в него ребер). Цель состоит в выборе множества узлов сети (вершин графа) S заданной мощности p

$$\arg\min_{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=p} f_{\mathcal{G}}(\mathcal{S}) = \arg\min_{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=p} \sum_{i\in\mathcal{V}} \min_{j\in\mathcal{S}} D(i,j). \tag{1}$$

Представим производственный график в виде трехмерной решетки в дискетной системе координат с осями i,k,l (время, производственные линиии, виды продукции). Каждый узел такой сети-решетки будем описывать тройкой его координат (i,k,l). Решению нашей задачи поставим в соответствие некоторое подмножество χ узлов нашей сети-решетки, задающих моменты переключения видов продукции на нашем графике. Задача состоит в выборе минимального по мощности множества χ узлов сети, удовлетворяющего заданным ограничениям.

Введем целочисленные переменные $y'_{i,k}$. Значение $y'_{i,k} = l$ будет означать выпуск l-го вида продукции на k-й линии в i-е сутки (значение $y'_{i,k} = 0$ – отсутствие выпуска). Пусть $y'_{0,k}, k = \overline{1,K}$ – целочисленные константы, имеющие значения от 0 до L, показывающие, на какой вид продукции настроена каждая из K линий в начальный момент времени. Для удобства описания введем дополнительные зависимые переменные $x'_{i,k}$

$$x'_{i,k} = \begin{cases} y'_{i,k}, & y'_{i,k} \neq y'_{(i-1),k}, \\ 0, & y'_{i,k} = y'_{i-1,k}, \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}.$$

Значения $y'_{i,k}$ могут быть получены из $x'_{i,k}$

$$y'_{i,k} = \begin{cases} y'_{(i-1),k}, & x'_{i,k} = 0, \\ x'_{i,k}, & x'_{i,k} \neq 0 \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}$$

и вспомогательные булевы переменные (в работе [2] задача сформулирована именно в булевых переменных)

$$y_{i,k,l} = \begin{cases} 1, & y'_{i,k} = l, \\ 0, & y'_{i,k} \neq l, \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}, l = \overline{1,L}.$$

Множества χ выбранных определяют значения переменных

$$x'_{i,k} = \begin{cases} l, & (i,k,l) \in \chi, \\ 0, & (i,k,l) \notin \chi. \end{cases}$$

Операция включения узла (i,k,l) в множество χ, таким образом, сводится с присвоению переменной $x'_{i,k}$ значения l, исключения – к присвоению значения 0.

Ограничения задачи сформулируем следующим образом:

$$f_2(\chi) = -\sum_{l=1}^{L} \max\{0, W_l - V_l \sum_{i=1}^{T_l} \sum_{k=1}^{K} y_{i,k,l}(3 - C'_{y'_{(i-1),k}, y'_{i,k}})\} = 0, \tag{2}$$

$$f_3(\chi) = -\sum_{i=1}^{I} \max\{0, W_{min} - \sum_{k=1}^{K} \sum_{l=1}^{L} V'_{y'_l}(3 - C'_{y'_{(i-1),k}, y'_{i,k}})\} = 0. \tag{3}$$

Здесь $C'_{l,r}$ – матрица $C_{l,r}$, дополненная нулевыми строкой и столбцом

$$C'_{l,r} = \begin{cases} C_{l,r}, & l > 0, r > 0, \\ 0, & r = 0, \\ 1, & l = 0, r > 0; \end{cases}$$

V' – вектор норм производства V, дополненный нулевым элементом V'_0=0.

Определим также целевую функцию $f(\chi) = |\chi|$.

Экономический смысл функции $f_2(\chi)$ – общее количество продукции, выпускаемой с отставанием от плана, функции $f_3(\chi)$ – суммарное недовыполнение суточного минимума выпускаемой продукции. Определим также единую штрафную функцию: $f_4(\chi) = f_2(\chi) + f_3(\chi)$.

2. Описание алгоритма

Идея генетического алгоритма с жадной эвристикой для *p*-медианной задачи на сети [5] состоит в следующем. Имеется некоторый массив (“популяция”) решений задачи, каждое из которых представляет собой множество из *p* выбранных узлов сети. Случайным образом выбираются два решения (родительские ”особи”). В качестве промежуточного решения принимается множество узлов, являющееся объединением выбранных “родительских” множеств, из которого по одному исключаются те узлы, удаление которых дает наименьший прирост целевой функции (1), пока не останется *p* узлов.

Наша задача, в отличие от *p*-медианной, – задача условной оптимизации. Задачу поиска решений, удовлетворяющих условиям (2) и (3), можно рассматривать как задачу минимизации функции $f_4(\chi)$. Характер генетического алгоритма не требует наличия единственного критерия оценки решений. Мы применим скоринг по трем критериям: значение целевой функции $f(\chi)$ и значения штрафных функций $f_2(\chi)$, $f_3(\chi)$. Кроме того, в p-медианной задаче число вершин *p* известно, в нашей задаче целью является его минимизация.

Общую схему алгоритма можно описать следующим образом.

Алгоритм 1. Генетический алгоритм с жадной эвристикой.

Шаг 1. Сгенерировать начальный массив множеств узлов сети, представленных тройками индексов (i, k, l): $A = \{\chi_j\} = \{(i_1, k_1, l_1), \dots, (i_{p_j}, k_{p_j})\}$, $j = \overline{1, N}$. Здесь N – число ”особей” популяции генетического алгоритма.

Шаг 2. Выбрать случайным образом два индекса родительских ”особей“ $j_1, j_2 \in \{\overline{1, N}\}$, $j_1 \neq j_2$ и индекс $j_3 \in w$. Здесь w – множество индексов ”особей“ (элементов массива A), оцениваемых как ”плохие” (см. ниже).

Шаг 3. Присвоить $\chi_{j_3} = \chi_{j_1} \cup \chi_{j_2}$.

Шаг 4. Выполнить процедуру мутации множества χ_{j_3} (опционально).

Шаг 5. Для каждого узла $\mathcal{V} = (i_1, k_1, l_1) \in \chi_{j_3}$ выполнять: если $\exists V_2 = (i_2, k_2, l_2) \in \chi_{j_3}: l_2 \neq l_1, i_2 = i_1, k_2 = k_1$, то выбрать случайным образом индекс $l \in \{l_1, l_2\}$, присвоить $\chi_{j_3} = \chi_{j_3} \backslash \{(i_1, k_1, l)\}$; следующая итерация цикла 5.

Шаг 6. Вычислить соответствующие множеству χ_{j_3} матрицы переменных $[y_{i,k,l}]$, $[x'_{i,k}]$, $[y'_{i,k}]$. Присвоить $FOUND = 0$.

Шаг 7. Разместив узлы множества χ_{j_3} в случайном порядке, для каждого узла $\mathcal{V} = (i', k', l') \in \chi_{j_3}$ выполнять:

Шаг 7.1. Присвоить $\xi = \chi_{j_3} \backslash \{\mathcal{V}\}$. Вычислить соответствующие множеству ξ матрицы булевых и целочисленных переменных $[y^{\xi}_{i,k,l}]$, $[x'^{\xi}_{i,k}]$ и $[y'^{\xi}_{i,k}]$. Если $f_4(\xi) < f_4(\chi_{j_3})$, то присвоить $\chi_{j_3} = \xi$, $FOUND = 1$, перерассчитать соответствущие χ_{j_3} матрицы переменных $[y_{i,k,l}]$, $[x'_{i,k}]$ и $[y'_{i,k}]$. Перейти к 7.3.

Шаг 7.2. Выполнить процедуры локального поиска в окрестности узла $\mathcal{V}$.

Шаг 7.3. Следующая итерация цикла 7.

Шаг 8. Если $FOUND = 1$, то присвоить *FOUND*=0 и начать цикл 7 заново.

Шаг 9. Проверить условия останова и перейти к шагу 2.

Мы использовали следующий алгоритм генерации начальной популяции, гарантирующий соответствие каждого решения ограничениям.

Алгоритм 2. Создание одной особи начальной популяции. Дано: число узлов сети p.

Шаг 1. Присвоить $x'_{i,k} = 0 \forall i = \overline{1, I}, k = \overline{1, K}$.

Шаг 2. Для $j = \overline{1, p}$ выполнять:

Шаг 2.1. Выбрать случайное $k \in \{\overline{1, K}\}$. Рассчитать $Z_k = \sum_{l=1}^{L} z_{k,l}$ – число видов продукции, которую можно производить на *k*-й линии. Выбрать случайное $m \in \{\overline{1, Z_k}\}$. Найти индекс m-го по счету ненулевого элемента k-го столбца матрицы $Z = [z_{k,l}]$, сохранить этот индекс в переменную l. Выбрать случайное $i \in \{\overline{1, T_l}\}$. Если $x'_{i,k} = 0$, то присвоить $x'_{i,k} = l$. Иначе начать шаг 2.1 сначала.

Шаг 2.2. Следующая итерация цикла 2.

Практика показывает, что наилучшие результаты достигаются, если p в различных экземплярах исходной популяции варьируется в широких пределах – в диапазоне $\{\overline{L, K \cdot I/2}\}$. Мы определяем его как $p = [L + j(K \cdot I/2 - L)/N]$, здесь j – номер генерируемого экземпляра; N – число экземпляров в популяции; $[\cdot]$ – целая часть числа. Эмпирическим путем установлено, что значение $N = I + K + L$ является достаточным, дальнейшее увеличение не приводит к улучшению результатов, лишь увеличивая время их достижения.

В качестве вспомогательных используются процедуры локального поиска в окрестности узла $\mathcal{V}$. Окрестностью в данном контексте будем называть множество узлов, отличающихся от $\mathcal{V}$ значением координаты l (соответствует замене вида продукции) либо координаты i .

Процедура 1. Выполняется для вершины $\mathcal{V} = (i, k, l)$ при условии $V_l \sum_{i=1}^{T_l} \sum_{k=1}^{K} y_{i,k,l}(3 - C'_{y'_{(i-1),k}, y'_{i,k}}) > W_l$. Составить множество индексов видов продукции S_L, удовлетворяющих $V_l \sum_{i=1}^{T_l} \sum_{k=1}^{K} y_{i,k,l}(3 - C'_{y'_{(i-1),k}, y'_{i,k}}) < W_l$. Выбрать случайным образом индекс l_2 из этого множества. Вычислить $\xi = (\chi_{j_3} \backslash \{\mathcal{V}\}) \cup \{(i, k, l_2)\}$. Если $f_4(\xi) < f_4(\chi_{j_3})$, то присвоить $\chi_{j_3} = \xi, FOUND = 1$.

Процедура 2. Выполняется для вершины $\mathcal{V} = (i, k, l)$ при условии $i > 1$ и $(i - 1, k, l) \notin \chi_{j_3}$. Вычислить $\xi = (\chi_{j_3} \backslash \{\mathcal{V}\}) \cup \{(i - 1, k, l)\}$. Если $f_4(\xi) < f_4(\chi_{j_3})$, то присвоить $\chi_{j_3} = \xi, FOUND = 1$.

Процедура 3. Выполняется для вершины $\mathcal{V} = (i, k, l)$ при условии $i < T_l (i + 1, k, l) \notin \chi_{j_3}$. Вычислить $\xi = \left(\chi_{j_3} \backslash \{\mathcal{V}\}\right) \cup \{(i + 1, k, l)\}$. Если $f_4(\xi) < f_4(\chi_{j_3})$, то присвоить $\chi_{j_3} = \xi, FOUND = 1$.

Процедура 4. Мутация множества χ_{j_3}. Выполняется с некоторой заданной вероятностью p_m. Сгенерировать множество ξ с помощью алгоритма 2 с параметром $p = |\chi_{j_3}|$. Присвоить $\chi_{j_3} = \chi_{j_3} \cup \xi$. Экспериментально установлены оптимальные значения $p_m \in [0{,}005;\ 0{,}03]$ при наличии процедур 2 и 3 и $p_m \in [0{,}01;\ 0{,}05]$ при наличии всех процедур локального поиска 1–3.

Процедура 5. Определение множества w методом скоринга.

Шаг 1. Присвоить $S_n = 0,\ F_n = |\chi_n|\ \forall n = \overline{1, N}$. Отсортировать массив F по возрастанию. Найти медианное значение $f' = F_{[N/2]}$.

Шаг 2. Присвоить $F_n = f_4(\chi_n) \forall n = \overline{1, N}$. Для $n \in \{\overline{1, N}\}: |\chi_n| \le f'$ присвоить $S_n = S_n + 1$. Отсортировать массив F в порядке возрастания. Найти медианное значение $f'_4 = F_{[N/2]}$.

Шаг 3. Для $n \in \{\overline{1, N}\}: f_4(\chi_n) \le f'_4$ присвоить $S_n = S_n + 2$.

Шаг 4. Для $n \in \{\overline{1, N}\}: f_2(\chi_n) = 0$ присвоить $S_n = S_n + 1$.

Шаг 5. Для $n \in \{\overline{1, N}\}: f_3(\chi_n) = 0$ присвоить $S_n = S_n + 1$.

Шаг 6. Найти наименьший индекс $n \in \{\overline{1, N}\}$: $f_4(\chi_n) = 0$ и $|\chi_n| = \min\limits_{q \in \{\overline{1,N}\}} |\chi_q|$. Если такой индекс существует, то присвоить $S_n = S_n + 10$.

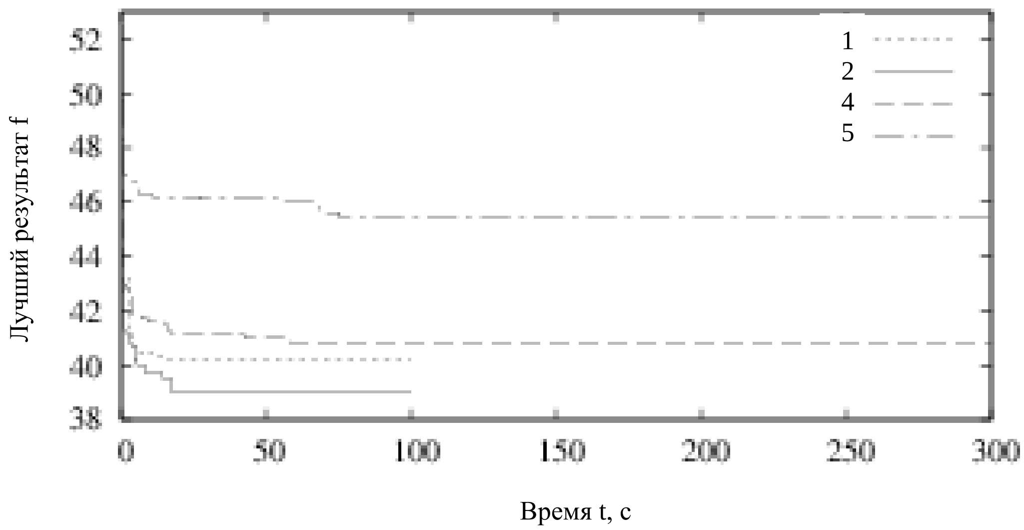
Шаг 7. Отсортировать массив S в порядке возрастания. Найти медианное значение $s' = S'_{[N/2]}$. Возвратить результат – множество $w = \{n \in \{\overline{1, N}\} | S_n \ge s'\}$.

Данная процедура требует достаточно больших вычислительных затрат, Поэтому процедура выполняется на шаге 2 после генерации начальной популяции, а также через каждые $[N/4]$ итераций.

3. Результаты

Алгоритм 2 реализован на языке Fortran 90 (компилятор ifort с опцией оптимизации кода и распаралеливания вычислений -O3). Для экспериментов использовалась вычислительная система Depo X8Sti (6-ядерное ЦПУ Xeon X5650 2.67 ГГц,12 Гб ОЗУ), технология hyperthreading отключена.

Получены результаты для задачи с параметрами, приведенными в работе [2] ($L = 34,\ K = 28,\ I = 31$), но с более жесткими ограничениями (2) и (3): $T_l \in \{\overline{10{,}31}\}$, $W_{min} = 1925$, $V_l \in [40{,}50]$, $W_l \in [20{,}25000]$. Всего использовано 10 наборов исходных данных, для каждого из них производилось по 10 запусков алгоритма в различных модификациях. На рисунке приведена зависимость наилучшего допустимого (т.е.$f_4(\chi) = 0$) решения от затраченного времени. Экспериментально выведено оптимальное для большинства случаев значение размера популяции *N=I+K+L*. Также даны результаты алгоритма локального поиска, составленного только из процедур 1, 2, 3. Параметры алгоритма: вероятность мутации $p_m = 0{,}01$.



Результаты и влияние размера популяции на результат: 1 – алгоритм 2 без процедур 1 – 3, $N = 89$; 2 – с процедурами 1–3, N$= 89$; 3 – без процедур 1–3, $N = 500$; 4 – с процедурами 1 – 3, $N = 500$; 5 – алгоритм локального поиска процедурами 1–3 с мультистартом

С учетом того, что время работы алгоритма, предложенного в работе [2], составляет $8 \cdot 10^5$ секунд, результаты, достигнутые в настоящей работе, можно считать убедительными.

Заключение. Задача календарного планирования загрузки производственных мощностей непрерывного производства может рассматриваться как дискретная задача размещения. Предложенный алгоритм на базе генетического алгоритма с жадной эвристикой для *p*-медианных задач решает поставленные задачи, при этом затрачиваемое на решение время на несколько порядков меньше, чем в случае применения существущего алгоритма псевдобулевой оптимизации.

Литература

1. *Фролов Е.* Оперативное планирование производства // Директор информационной службы. – 2013. – Вып. 5. – URL: http://www.osp.ru/cio/2013/05/13035711/ (дата обращения: 01.10.2013).
2. *Antamoshkin A., Masich I.* Pseudo-Boolean Optimization in Case of an Unconnected Feasible Set, in: "Models and Algorithms for Global Optimization" // Optimization and Its Applications. – 2007. – V. 4. – P.111–122.
3. *Kazakovtsev L.A., Gudyma M.N., Antamoshkin A.N.* Genetic Algorithm with Greedy Heuristic for Capacity Planning // 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). – S.-Petersburg, 2014. – 6–8 October. – P. 607–613.
4. *Avella P., Sassano A., Vasil'ev I.* Computational Study of Large-Scale p-Median Problems // Mathematical Programming. – 2007. – Issue 109(1). – P. 89–114.
5. *Antamoshkin A.N., Kazakovtsev L.A.* Random Search Algorithm for the p-Median Problem // Informatica. – 2013. – V. 37(3). – P. 267–278.
6. *Alp O., Erkut E., Drezner Z.* An Efficient Genetic Algorithm for the p-Median Problem // Annals of Operations Research. – 2003. – V.122(1–4). – P. 21–42.