Методы решения задачи линейного раскроя с минимизацией перестановок ножей

В.В. Клименко, Л.В. Щеголева

Петрозаводский государственный университет

Аннотация: В данной статье представлены методы решения задачи линейного раскроя (ЗЛР) с критерием минимизации числа отходов и перестановок ножей. Задача линейного раскроя в общем виде представляет собой оптимизационную задачу, которая заключается в размещении заданных видов материала (рулонов) так, чтобы минимизировать отходы и/или максимизировать использование исходных материалов с учетом ограничений по количеству ножей, ширины тамбура и требуемых заказов. Рассматривается частный случай задачи с дополнительным условием по минимизации перестановок ножей и следующие подходы для его решения: метод полного перебора, а также случайный поиск на основе генетических и эволюционных алгоритмов. Для различных методов решения ЗЛР представлен псевдокод. Проведено сравнение по алгоритмической сложности, контролируемости времени исполнения и точности. Случайный поиск на основе генетических и эволюционных алгоритмов оказался более приспособленным для решения задачи.

Ключевые слова: планирование производства бумаги, линейный раскрой, полный перебор, генетический алгоритм.

Обзор задачи линейного раскроя

Целлюлоза и древесная масса отправляются в смесительные бассейны, где они смешиваются и превращаются в однородную бумажную массу. Эта масса затем под давлением поступает на бумагоделательную машину (БДМ), где начинается процесс производства бумажного полотна. Технологический процесс включает несколько этапов [1]: сначала происходит обезвоживание бумажной массы на сеточном столе, затем в прессовой части, после чего полотно сушится и каландрируется для придания ему нужной текстуры и плотности. Когда бумажное полотно достигает необходимой характеристики, оно наматывается на тамбурный вал. С помощью крана полотно снимается с наката БДМ и переносится на продольно-резательные станки, где происходит его нарезка на рулоны заданных форматов.

Процесс нарезки бумаги включает в себя несколько ключевых этапов [2]. Сначала бумажное полотно подается на резальные машины, где оно

аккуратно выравнивается и фиксируется. Затем, с помощью острых ножей или резальных дисков, полотно разделяется на рулоны нужной ширины. Эти станки могут быть настроены на различные форматы, что позволяет производить рулоны различной ширины в зависимости от заказа. После нарезки готовые рулоны перемещаются по транспортерам к упаковочной линии. Здесь они проходят финальную проверку качества и упаковку в оберточную бумагу, что обеспечивает защиту рулонов во время хранения и транспортировки.

Однако, нарезка бумажного полотна должна быть организована таким образом, чтобы минимизировать отходы и обеспечить максимальную эффективность использования материала. Это приводит к необходимости математической формулировки задачи линейного раскроя (ЗЛР), где необходимо оптимально распределить рулоны в соответствии с шириной форматов, чтобы достичь наилучших результатов в процессе нарезки, учитывая ограничения на ширину исходного полотна (тамбура) и требования к конечной продукции [3].

Пусть:

– L — ширина тамбура БДМ

– $M = \{(l_1, q_1), (l_2, q_2), \dots, (l_m, q_m)\}$ — множество заказов, где каждый заказ i имеет ширину формата l_i и количество рулонов заказов q_i;

– $p = (p_1, p_2, \dots, p_v)$ — раскрой, представляющий собой упорядоченную последовательность, где для каждого p_j существует такой заказ $(l_k, q_k) \in M$, что $p_j = l_k$, при этом количество элементов в разных раскроях может быть разным;

– $P = \{p^1, p^2, \dots, p^a\}$ — план раскроя, то есть упорядоченная последовательность раскроев.

Необходимо найти такой план раскроя P, для которого суммарная ширина отходов будет минимальной [4]. Целевая функция τ суммы отходов задается выражением:

$$\tau(P) = \sum_{i=1}^{a}(L - \sum_{j=1}^{v_i} p_j^i) \to min.$$

Заданы следующие ограничения:

$\sum_{j=1}^{v} p_j < L, \forall p \in P;$ (1)

$\sum_{i=1}^{a}\sum_{j=1}^{v_i} I(p_j^i = l_k) = q_k, \ \ k \in [1, m].$ (2)

Ограничение (1) накладывает условие на раскрой: сумма форматов не должна превышать ширину тамбура БДМ. Ограничение (2) описывает условие выполнения всех заказов.

Однако для повышения эффективности работы предприятия можно добавить в задачу требование на минимизацию количества перестановок ножей для искомого плана раскроя [5]. План раскроя должен быть минимальным, в первую очередь, по числу отходов, а потом минимальным по числу перестановок ножей.

Пусть, $\sigma(p^1, p^2) = v_2 - \max_k\{0 \le k \le min\{v_1, v_2\}: \forall j \le k \ \ p_j^1 = p_j^2\}$ – функция расчета перестановок ножей для перехода от раскроя p^1 к раскрою p^2. Она равна разности между количеством форматов в раскрое p_2 и количеством совпадающих форматов в начале упорядоченной последовательности форматов для раскроев p^1 и p^2. Совпадающие в начале последовательности форматы не требуют перестановки ножей, поэтому их можно не учитывать.

Функцию расчета числа перестановок ножей для плана раскроя P можно записать следующим образом: $\mu(P) = v_1 + \sum_{i=1}^{a-1}\sigma(p^i, p^{i+1})$. Эта сумма состоит из количества форматов в начальном раскрое p^1, так как изначально

ножи не установлены. Также сумма включает в себя число необходимых перестановок ножей для каждой пары раскроев p^i и p^{i+1}.

Тогда задача минимизации числа перестановок ножей БДМ в раскрое P для ЗЛР с минимизацией отходов будет следующей: $\mu(P) \to min$ при условии, что на множестве P достигается минимум функции $\tau(P)$.

Метод полного перебора

Метод полного перебора представляет собой исчерпывающий подход к решению ЗЛР, обеспечивая нахождение глобально оптимального решения. Однако при наличии высоких структурных сложностей, присущих задачам линейного раскроя, этот метод сталкивается с существенными ограничениями из-за своей вычислительной сложности, столь высокой, что она порой делает его неразумным для практического применения [6]. Алгоритмическая сложность для данного метода будет экспоненциальной от количества заказов [7].

Предлагается один из вариантов полного перебора с использованием рекурсии. Данный алгоритм находит план раскроя минимальный по числу отходов и по числу перестановок ножей. На каждой итерации он рассматривает текущий план раскроя и возвращает наилучший доступный вариант. Изначально рассматриваемый план раскроя пустой. Алгоритм можно разделить на 2 основные части: в первой части проверяется, что текущее решение удовлетворяет ограничению (2) и тогда его можно вернуть как лучшее на данной итерации; вторая часть выполняется, если текущее решение не удовлетворяет ограничению (2) и, соответственно, необходимо добавить еще форматы в план раскроя. На каждой итерации алгоритма создается новый план раскроя и записывается в переменную newP. При этом новый формат помещается в последний раскрой (переменная LastCut), если это не нарушает ограничение (1), иначе добавляется новый раскрой (переменная newLastCut). Для нового плана раскроя вызывается новая

рекурсивная итерация перебора, полученный в результате ее выполнения план раскроя (переменная NewBestP) сравнивается с текущим наилучшим планом раскроя (переменная BestP). Текущий наилучший план раскроя будет заменен на план из переменной NewBestP в 3 случаях: – если в переменной BestP находится пустой план раскроя; – если план раскроя в переменной NewBestP лучше плана раскроя в переменной BestP по числу отходов; – или если план раскроя в переменной NewBestP совпадает по числу отходов, но лучше по числу перестановок ножей. После рассмотрения всех возможных вариантов добавления заказов возвращается наилучший план раскроя из переменной BestP. Псевдокод алгоритма рекуррентного перебора представлен ниже. Вход: множество заказов $M = \{(l_1, q_1), (l_2, q_2), \dots, (l_m, q_m)\}$; ширина тамбура L; текущий план раскроя P, при первом вызове – это пустой план раскроя. Выход: план раскроя в переменной $BestP$, минимальный по числу отходов и перестановок ножей.

$EnumeratePlanCuts(M, L, P = \emptyset)$:

```
BestP = ∅
# Счетчик форматов заказов для текущего плана раскроя
MCount = {l_1: 0,  l_2: 0, … , l_m: 0}
for p in P
        for format in p
                # Подсчитываем количество форматов во всем плане раскроя
                MCount[format]++
# Проверка, что текущий формат выполняет заказ
orderCompleted = 1
for i = 1 to m
        if M[i].q != MCount[M[i].l]
                # В случае если заказ не выполнен
```

orderCompleted = 0

if orderCompleted = 1

 # Если план раскроя содержит необходимое число заказов, то к нему нельзя добавить новые форматы, поэтому его можно вернуть как лучший вариант

BestP = P

return BestP

for i = 1 to m

if M[i].q < MCount[M[i].l]

 # количество планов раскроя

PQuantity = $|P|$

 # последний раскрой в текущем плане раскроя

LastCut = P[PQuantity]

 # суммарная ширина последнего раскроя

LastCutLen = $\sum_{j=1}^{LastCut} LastCut[j]$

 # Если добавление нового формата в последний раскрой не нарушает ограничение по максимальной ширине раскроя

If LastCutLen + M[i].l <= L

newLastCut = LastCut + {M[i].l}

newP = P

 # Добавляем в последний раскрой новый формат

newP[PQuantity] = newLastCut

Else

 # Добавляем раскрой из одного формата в новый план раскроя

newP = P + {M[i].l}

 # Рассматриваем новый план раскроя

NewBestP = $EnumeratePlanCuts(M, L, newP)$

 # Проверка, что новый план раскроя является лучшим вариантом

if NewBestP != ∅

 # Если еще не выбран лучший план раскроя

if BestP = ∅

BestP = NewBestP

 # Когда новый план раскроя меньше по числу отходов

if $\tau(BestP) > \tau(NewBestP)$

BestP = NewBestP

 # Когда новый план раскроя равен по числу отходов, но меньше по числу перестановок ножей

If $\tau(BestP) = \tau(NewBestP)\ and\ \mu(BestP) > \mu(NewBestP)$

BestP = NewBestP

return BestP

Генетический поиск

Генетические и эволюционные алгоритмы (ГА и ЭА) представляют собой методы поиска и оптимизации, которые могут существенно улучшить процессы решения ЗЛР с минимизацией перестановок ножей. Одним из основополагающих преимуществ этих алгоритмов является возможность гибкого контроля времени выполнения, что позволяет адаптировать их под требования конкретной задачи. При этом каждая итерация не ухудшает качество текущего решения, что делает процесс наименее уязвимым для локальных минимумов [8].

Эффективность применения ГА и ЭА тесно связана с мощностью вычислительных ресурсов, что нередко предполагает использование

высокопроизводительных вычислительных систем, таких как суперкомпьютеры и кластеры [9].

Предлагается следующий алгоритм случайного поиска с элементами генетического алгоритма [10]. Назовем его генетическим поиском (ГП). Он итеративно образует новые популяции $S = \{S_1, S_2, \dots, S_C\}$, где S_i – это i-ый план раскроя P, а S_{ij} – это j-ый раскрой p в i-ом плане раскроя P. Изначальная популяция S_0 состоит из случайных планов раскроя, которые удовлетворяют ограничениям (1)–(2). Размер начальной популяции определяет число итераций, соответственно влияет на точность и скорость исполнения алгоритма.

Каждая итерация начинается с мутации (функция $Mutate$) для планов раскроев текущей популяции S_t. Функция $Mutate$ создает план раскроя P', и если он лучше старого плана раскроя, то есть $\tau(P) > \tau(P')$, то план раскроя заменяется $P = P'$. После этого выполняется кроссинговер (функция $Crossingover$) для пары соседних планов раскроя $S_{t,i*2-1}$ и $S_{t,i*2}$. Функция $Crossingover$ возвращает один лучший из двух планов раскроя, тем самым образует в новой популяции план раскроя $S_{t+1,i}$. Это уменьшает размер популяции S_{t+1} в 2 раза по сравнению с S_t. По итогу работы алгоритма популяция сокращается до одного плана раскроя P', который будет не хуже планов раскроя из начальной популяции $P \in S_0$. Чем больше размер изначальной популяции, тем больше итераций и, соответственно, выше вероятность получить план раскроя P' с минимальным числом отходов и перестановок ножей.

Пусть t – это номер итерации поиска и популяции.

S – это множество планов раскроя P, которые соответствуют ограничениям (1)–(2), иначе говоря, популяция планов раскроя. А S_t – это популяция на итерации поиска t.

C – размер текущей популяции.

$Mutate(P)$ – это функции мутации, которая случайным образом перемешивает раскрои $p \in P$ и порядок форматов внутри них. Функция $shuffle$, случайно перемешивает порядок форматов раскроя p. Использование функции $Mutate$ повышает вероятность получить улучшенный план раскроя с точки зрения минимизации числа перестановок ножей. Вход функции: планы раскроя P. Выход: план раскроя с измененным порядком P'.

$Mutate(P)$:

$P' = P$

for i = 1 to a

Перемешать раскрой P'_i

$shuffle(P'[i])$

$shuffle(P')$

return P'

$Crossingover(P^1, P^2)$ – это функция кроссинговера для двух планов раскроя P^1 и P^2. Эта функция выбирает лучший из планов на основе целевой функции. Использование функции $Crossingover$ позволяет избавиться в популяции от не самых лучших планов. Вход функции: планы раскроя P^1, P^2. Выход: лучший из обоих планов раскроя.

$Crossingover(P^1, P^2)$:

if $\tau(P^1) > \tau(P^2)$

return P^2

else return P^1

Алгоритм генетического поиска представлен ниже. Вход: S_0 – начальная популяция планов раскроя. Выход: итоговый планы раскроя P'.

$GeneticSearch(S_0)$:

t = 0

while $C_t > 1$

$n = C_t$

for i = 1 to n

$P' = Mutate(S_{t+1,i})$

 # Сохранение наилучшего результата

if $\tau(S_{t,i}) > \tau(P')$

$S_{t,i} = P'$

 # Запись в следующую популяцию результата кроссинговера

for i = 1 to $\left\lfloor \frac{n}{2} \right\rfloor$

$S_{t+1,i} = Crossingover(S_{t,i*2-1}, S_{t,i*2})$

 # Если размер популяции нечетный, то перенесем оставшийся элемент

if n % 2 = 1

$S_{t+1,i} = S_{t,\lceil n/2 \rceil}$

t++

$P' = S_{t,1}$

return P'

Всего ГП выполнит $O(\log(|S_0|))$ итераций, так как размер популяций каждый раз уменьшается в 2 раза с округлением вверх. Но каждая итерация содержит обработку планов раскроев в функциях $Mutate$, $Crossingover$ и τ. План раскроя состоит из $\sum_{i=1}^{|M|} q_i$ форматов, так как должен удовлетворять ограничению (2). Тогда, итоговая алгоритмическая сложность генетического поиска равна $O(\sum_{i=1}^{|M|} q_i \ \log(|S_0|))$.

Сравнение алгоритмов

Алгоритм перебора имеет экспоненциальную алгоритмическую сложность от числа заказов, тогда как ГП – $O(\sum_{i=1}^{|M|} q_i \ \log(|S_0|))$. Очевидно,

что ГП быстрее полного перебора, однако скорость достигается за счет меньшего рассмотрения возможных вариантов.

Метод полного перебора вынуждает перебрать все возможные варианты. Тогда как для ГП можно задать количество итераций через размер начальной популяции, что позволяет контролировать время исполнения.

Также метод полного перебора гарантирует рассмотрение всех вариантов и, таким образом, находит наилучший план раскроя. ГП позволяет лишь найти план раскроя P', который будет не хуже изначальной популяции S_0, но увеличение размера начальной популяции повышает вероятность найти более хороший план раскроя.

Заключение

В статье рассмотрен частный случай ЗЛР с минимизацией числа отходов и перестановок ножей. Для данного случая предложены 2 алгоритма нахождения наилучшего решения: метод полного перебора и генетический поиск (ГП). Метод полного перебора гарантирует нахождение наилучшего решения. Однако он намного медленнее, чем ГП. ГП более быстрый и позволяет регулировать время исполнения, но не гарантирует, что ответ будет наилучшим. Таким образом, ГП более приспособлен для решения задачи.

Данный алгоритм в дальнейшем планируется рассмотреть для ЗЛР с несколькими БДМ.

Литература

1. Ромолдович У. А., Кузнецов В. А. Математические модели и методы учета сроков продукции в задаче раскроя тамбуров бумагоделательных машин //Ученые записки Петрозаводского государственного университета. – 2014. – №. 4 (141). – С. 112-115.

2. Koch S., König S., Wäscher G. Integer linear programming for a cutting problem in the wood-processing industry: a case study //International Transactions in Operational Research. – 2009. – V. 16. – №. 6. – T. 715-726.

3. Мухачева Э. А., Мухачева А. С. ЛВ Канторович и задачи раскроя-упаковки: новые подходы для решения комбинаторных задач линейного раскроя и прямоугольной упаковки //Записки научных семинаров ПОМИ. – 2004. – Т. 312. – №. 0. – С. 239-255.

4. Картак В. М. Решение задачи линейного раскроя с помощью метода группировки //Вестник Башкирского университета. – 2005. – Т. 10. – №. 3. – С. 22-25.

5. Martin M., Yanasse H. H., Salles-Neto L. L. Pattern-based ILP models for the one-dimensional cutting stock problem with setup cost // Journal of Combinatorial Optimization. – 2022. – V. 44. – №. 1. – pp. 557-582.

6. El-Dessouki O. I., Huen W. H. Distributed enumeration on between computers //IEEE Transactions on Computers. – 1980. – V. 29. – №. 09. – pp. 818-825.

7. Cung V. D., Hifi M., Le Cun B. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm // International Transactions in Operational Research. – 2000. – V. 7. – №. 3. – pp. 185-210.

8. Подлазова А. В. Генетические алгоритмы на примерах решения задач раскроя // Проблемы управления. – 2008. – №. 2. – С. 57-63.

9. Easwaran A. M., Drossopoulou S. A Parallel Genetic Algorithm Approach To The Knife Change Minimisation Problem // The Proceedings of the sixth Parallel Computing Workshop (PCW'96), Japan. – 1996.

10. Воронов Р.В., Шабаев А.И., Клименко В.В. Генетический алгоритм для задачи линейного раскроя с допусками на объемы выпуска продукции // Программная инженерия. - Москва, 2024. -№1. - С.35-43. (ВАК, RSCI, РИНЦ)

References

1. Romoldovich U. A., Kuzneczov V. A. Ucheny`e zapiski Petrozavodskogo gosudarstvennogo universiteta. 2014. №4. pp. 112-115.

2. Koch S., König S., Wäscher G. International Transactions in Operational Research. 2009. Vol.16. №6. pp. 715-726.

3. Muxacheva E`. A., Muxacheva A. S. Zapiski nauchny`x seminarov POMI. 2004. Vol. 312. №0. pp. 239-255.

4. Kartak V. M. Vestnik Bashkirskogo universiteta. 2005. Vol. 10. №3. pp. 22-25.

5. Martin M., Yanasse H. H., Salles-Neto L. L. Journal of Combinatorial Optimization. 2022. Vol. 44. №1. pp. 557-582.

6. El-Dessouki O. I., Huen W. H. IEEE Transactions on Computers. 1980. Vol. 29. №09. pp. 818-825.

7. Cung V. D., Hifi M., Le Cun B. International Transactions in Operational Research. 2000. Vol. 7. №3. pp. 185-210.

8. Podlazova A. V. Problemy` upravleniya. 2008. №2. pp. 57-63.

9. Easwaran A. M., Drossopoulou S. Proceedings of the sixth Parallel Computing Workshop 1996. P2. pp. W1-W13

10. Voronov R.V., Shabaev A.I., Klimenko V.V. Programmnaya inzheneriya. 2024. №1. pp. 35-43.

Дата поступления: 29.12.2024

Дата публикации: 25.02.2025