МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВПО «СЫКТЫВКАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ПИТИРИМА СОРОКИНА»
ИНСТИТУТ ТОЧНЫХ НАУК И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ В ОБРАЗОВАНИИ

УТВЕРЖДАЮ
Зав. кафедрой прикладной математики,
к.ф.-м.н., доцент,
_____Ермоленко А. В.
«____»_____ 2016г.

ОТЧЁТ

об учебной практике
в Сыктывкарском государственном университете
в период с 27.06.2016 по 9.07.2016
Тема: «Генетические алгоритмы. Итеративный способ решения задачи раскроя»

Научный руководитель
к.ф.-м.н., доцент:
_____Н. О. Котелина
«____»_____ 2016г.

Исполнитель, студент
145 группы:
_____Мельников Вадим
«____»_____ 2016г.

Сыктывкар 2016

Содержание

1. Введение **3**

2. Основные характеристика задач раскроя **4**

2.1. Пространственные характеристики 4

2.2. Количественные характеристики 4

2.3. Геометрические характеристики 5

2.4. Характеристики по ограничениям на результат 5

3. Постановка задачи **6**

4. Генетические алгоритмы **8**

4.1. Кодирование задачи 8

4.2. Инициализация первого поколения 9

4.3. Оценка индивидов 9

4.4. Скрещивание 10

4.5. Мутация 11

4.6. Отбор 11

5. Заключение **13**

Приложение 1. Исходный код функций ГА **15**

Список литературы **20**

1. Введение

Экономия материала представляет собой сложную и важную проблему, с которой часто приходится встречаться на различных производствах, при резке различных материалов на: листы металла, стекла или дерева, трубы, профильный прокат, изделия сложной формы. Для её решения необходимо максимизировать использование материала, из которого вырезаются заготовки, что по сути и является рациональным раскроем материала. Максимизация использования материалов позволяет достичь большой экономии денежных средств.

На самом деле, задача раскроя является NP-полной даже для прямоугольников. Для фигур неправильной формы геометрическая сложность увеличивает количество совершаемых вычислений, поэтому применяются различные эвристические методы решения задачи.

2. Основные характеристика задач раскроя

Прежде чем приступать к рассмотрению алгоритмов решения задачи раскроя, следует рассмотреть характеристики, влияющие на то, как будет выглядеть итоговый алгоритм решения. В статье [1] Harald Dyckhoff приводит достаточно полное описание характеристик задач раскроя.

2.1. Пространственные характеристики

Основная характеристика раскроя — количество измерений:

— раскрой в одномерном пространстве;

— раскрой в двумерном пространстве;

— раскрой в трёхмерном пространстве.

Например загрузка поддонов является задачей в двумерном пространстве. В отличие от задач в двух и более измерениях, задача в одномерном пространстве имеет явное решение. Достаточно подробно данная задача описывается в книге [2] Канторовича-Залгаллера — «Рациональный раскрой промышленных материалов». Так же в данной книге можно найти методы решения задач двумерного раскроя для случая прямоугольнных заготовок.

2.2. Количественные характеристики

Другая важная характеристика — количественная. В задаче раскроя необходимо некоторым образом измерять количественные и качественные характеристики фигур. Например, площадь, длина и ширина фигур. Или количество уже расположенных фигур. Тут можно рассмотреть два варианта:

— дискретное измерение с помощью, например, натуральных и целых чисел;

— дробное измерение на основе вещественных чисел.

Первый вариант позволяет нам подсчитывать количество изделий, уже расположенных на материале, а с помощью второго можно измерять различные характеристики фигур, такие как площадь, длина и ширина.

2.3. Геометрические характеристики

Не малую роль играют в раскрое сами фигуры, которые необходимо расположить на плоскости. В пространстве фигуры однозначно определяются с помощью следующих свойств:

— формой;

— размером;

— ориентацией;

— правильной или неправильной формой.

2.4. Характеристики по ограничениям на результат

По ограничениям на результат можно выделить четыре основные группы:

— минимальное расстояние между объектами;

— ориентация фигур относительно друг друга;

— ограничение на количество фигур;

— ограничение на количество совершаемых «резов».

Автор выделяет ещё несколько групп по различным признакам, но данные являются основными.

3. Постановка задачи

Прежде чем переходить к алгоритмам, применяемым для решения задачи раскроя, следует рассмотреть математическую постановку задачи. Задача ставится по аналогии с задачей раскроя в одномерном пространстве [3].

Имеется сырьё площади S, на котором неоьходимо расставить заготовки M различных типов. Площади заготовок задаются вектором $s[M]$, а количество заготовок каждого типа задаётся вектором $b[M]$.

Пусть матрица $A[M, N]$ — целочисленная матрица всех потенциально возможных способов раскроя одной единицы сырья, $|N| = n$ — число таких способов. Потенциально возможный способ раскроя — столбец матрицы $A[M, j], j \in 1 : N$, удовлетворяющий условию:

$$s[M] \cdot A[M, j] \leq S. \tag{1}$$

Пусть $x[M]$ — искомый вектор, являющийся некоторым столбцом матрицы A. Тогда задача минимизации суммарных отходов сырья запишится следующим образом:

$$\begin{cases} f = S - x^T[M] \cdot s[M] \to \min \\ x[M] \leq b[M] \\ S - x^T[M] \cdot s[M] \geq 0 \\ x[M] \geq 0 \end{cases} \tag{2}$$

Важной особенностью задачи раскроя в двумерном пространстве является то, что не каждый столбец матрицы A удовлетворяющий заданным условиям является реальным решением. Данный факт объясняется произвольностью форм заготовок, поэтому нельзя одназначно утверждать, что если заданные условия выполнены для некоторой последовательности, то эта последовательность является решением.

Из сказанного выше следует нелинейность задачи и то, что решение задачи может быть не только на границе многогранника, заданного условиями. Таким образом, необходимо отыскивать последовательность с помощью некоторого «оптимизированного» перебора. Также заданные условия никак не определяют координаты расстановки заготовок на единице сырья. Методы для решения данныъ проблем и будут описываться в дальнейшем.

4. Генетические алгоритмы

Таким же образом, как нейронные сети пытаются имитировать мощь мозга, эволюционные алгоритмы (ЭА) воспроизводят биологическую эволюцию. Например, природа создала сложную структуру человеческого мозга из простейших элементов, пользуясь лишь мощью эволюции и естественного отбора. Так же как естественный отбор действует на популяции животных, позволяя лучшим выжить и передать свои гены потомкам, так же и ЭА действует, например, на последовательности раскроя, нейронные сети, электрические контуры и так далее. Они определяют качество каждого элемента в популяции, позволяя лучшим выжить и убивая слабейших. Далее — очень важный шаг — они позволяют различным решениям «скрещиваться» между собой, порождая, возможно, более жизнеспособные решения.

Генетический алгоритм (ГА) — самый популярный из эволюционных алгоритмов. Он был изобретён Джоном Холландом в Университете Мичигана в 1975 году. По изначальной задумке, ГА использовал только бинарные числа, но для извлечения большей пользы будем пользоваться десятичными числами. Сам алгоритм состоит из нескольких частей [5]:

1. Кодирование проблемы в виде генов.
2. Выведение первоначального поколения.
3. Вычисление оценок для индивидов.
4. Скрещивание.
5. Мутация.

Рассмотрим ГА на применительно к задаче раскроя [6].

4.1. Кодирование задачи

Основываясь на предложенных ранее методах представления и перемещения фигур, можно заметить, что не имеет смысла включать в геном

координаты положения и угол поворота так, как они будут вычислены при расположении последовательности. Достаточно кодировать лишь саму последовательность раскроя. Отметим, что для достижения результата, в геноме инидвида не должно быть повторов, соответственно каждый ген должен быть уникален. Таким образом индивидом будет являться — последовательность просто раскроя, ведь углы постановки и координаты будут восстановлены при расположении, а геном — порядковый номер фигуры в исходном наборе — это позволит избежать повторов.

4.2. Инициализация первого поколения

Первое поколение индивидов оказывает огромное влияние на результат всего ГА. Чтобы получить неплохой первичный набор, можно расположить по порядку отсортированные по площади фигуры, а остальных индивидов первого поколения получить с помощью мутации.

4.3. Оценка индивидов

Оценку результата можно проводить различными способами. Можно, к примеру, ввести некоторую оценочную функцию. Рассматривая задачу раскроя, за оценку можно взять занятую фигурами площадь. Далеко необязательно использовать какой-то один фактор, например, если индивид I характеризуется парой (n, h), где n — число размещённых фигур, а h — высота раскроя, то можно взять за оценку данную пару. Тогда то, что индивид I_1 лучше, чем индивид I_2 можно записать через следующее отношение ρ:

$$I_1 \ \rho \ I_2 = (n_1 > n_2 \vee (n_1 = n_2 \wedge h_1 < h_2)). \tag{3}$$

Тоже самое можно записать с помощью следующей функции оценки индивидов:

$$f(I) = n + h^{-1}, \tag{4}$$

тогда достаточно сравнивать числовые значения функций. Выбранная функция будет принимать большее значение при лучшем варианте рас-

кроя. Вообще говоря, можно использовать любую функцию, исходя из условий задачи.

4.4. Скрещивание

Рассмотрим скрещивание на следующем примере:

1. Выберем двух индивидов, например [5 2 3 7 6 1 4] и [4 6 2 1 3 5 7].
2. Случайным образом выберем часть первого родителя, которая перейдёт в потомка: [5 2 (3 7) 6 1 4].
3. Удалим эти элементы из второго родителя: [4 6 2 1 (3) 5 (7)].
4. Первого потомка получим путём копирования генов второго родителя в первого: [4 6 3 7 2 1 5].
5. Повторим действия 1—3, поменяв родителей местами.
6. Второго потомка получим путём копирования генов первого родителя во второго: [5 3 2 1 7 6 4].

Из курса математического анализа известно, что функция может иметь некоторое количество точек экстремума. Минимальное значение может достигаться в нескольких различных точках, но при этом есть и экстремумы которы минимумом не являются.

В процессе скрещивания стоит использовать не только самых лучших, индивидов но и достаточно хороших в целях избежания попадания на экстремумы, не являющиеся точками минимума локальные оптимумы. Данное явление называется преждевременной сходимостью или сходимостью к квазиоптимальному решению [7].

На рис. 1, для некоторой функции $f : X \to Y$, продемонстрированы точки квазиоптимального и оптимального решения. На рисунке цифра 1 — квазиоптимальное решение, а 2 — оптимальное.

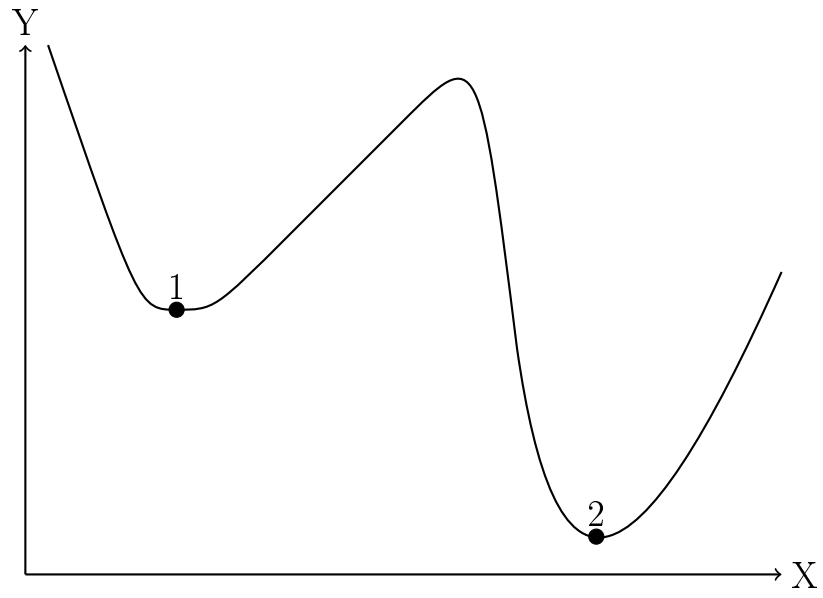

Рис. 1. Преждевременная сходимость

4.5. Мутация

Мутация происходит после скрещивания и применяется к новым индивидам, на самом деле можно применять и к появившимся ранее. В случае раскроя в процессе мутации меняются местами два гена в последовательности. Вероятность мутации должна быть не очень высокой, приблизительно 10%, также можно изменять экспериментально. Иногда можно переставлять не просто два гена, а целые части последовательности, но не стоит делать этого слишком часто.

4.6. Отбор

Теперь, когда для каждого индивида вычислена оценка, можно провести отбор для создания нового поколения. Самый простой вариант — взять только те индивиды, у которых достаточно хорошая оценка. В классическом варианте генетического алгоритма, индивиды в новом поколении выбираются случайно, а вероятность попадания индивида в новое поколение будет пропорциональна его оценке. Например,

$$p_i = f_i^{-1} \sum f_k. \tag{5}$$

На самом деле, включать в новое поколение можно не только те индивиды, которые были выведены на данном шаге, но и те которые уже были ранее, ведь они могут быть не хуже чем те, что только что появились.

5. Заключение

В результате было реализовано два метода раскроя. Один на основе фигур в виде многоугольников, другой использует растровые матрицы. Первый показал свою полную несостоятельность. Он имеет огромную вычислительную сложность при очень низком качестве на реальных примерах. В дальнейшем развиваться будет только растровый метод. Он отлично подходит для фигур с большим количеством вершин и сложными формами.

Для наглядности результата, возьмём лист 1063×1063 мм. В раскройный план входят следующие наименования фигур:

1. «Олень» — 5 штук, с шагом 15 градусов.
2. «Обезьянка» — 5 штук, с шагом 15 градусов.
3. «Голубь» — 5 штук, с шагом 15 градусов.
4. «Лошадка» — 5 штук, с шагом 15 градусов.
5. «Ангелок» — 5 штук, с шагом 15 градусов.

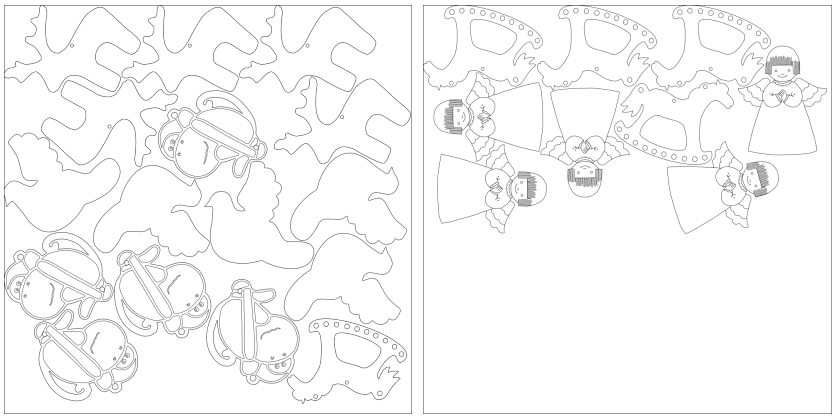

Рис. 2. Первый и второй раскройные листы

За 10 итераций генетического алгоритма были достигнуты следующие результаты:

1. Раскройный план полностью выполнен с использованием двух листов.

2. Первичная высота раскроя на первом листе — 298.3 мм, итоговая высота — 292.1 мм.

3. Первичная высота раскроя на втором листе — 172.2 мм, итоговая высота — 167.4 мм.

Следует отметить, что для примера взят достаточно маленький лист. Экономия материала линейно возрастает с увеличением площади. Результаты можно увидеть на рис. 2.

Приложение 1. Исходный код функций ГА

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <float.h>
#include <math.h>
#include "figure.h"
#include "nest_structs.h"
#include "crosscheck.h"
#include "cmnfuncs.h"
#include "nestdefs.h"

int mutate(struct Individ *src, struct Individ
  *mutant, int setsize);
int gensequal(struct Individ *indiv1, struct Individ
  *indiv2);
int gensequal2(struct Individ *indiv1, struct Individ
  *indiv2, struct Figure *figset);
int crossover(struct Individ *par1, struct Individ
  *par2, struct Individ *child, int setsize);

int crossover(struct Individ *par1, struct Individ
  *par2, struct Individ *child, int setsize)
{
        int i, j;
        int g1, g2;

        if (par1->gensize < 3)
                return SMALL_INDIVID;
```

```
if (par1->gensize != par2->gensize)
        return DIFFERENT_SIZE;

srand(time(NULL));

g1 = rand() % (par1->gensize);
g2 = rand() % (par2->gensize);

while (g1 == g2)
        g2 = rand() % (par1->gensize);

child->par1 = par1->genom;
child->par2 = par2->genom;
child->gensize1 = par1->gensize;
child->gensize2 = par2->gensize;
child->g1 = g1;
child->g2 = g2;

child->genom = (int*)xmalloc(sizeof(int) *
   setsize);
child->gensize = par1->gensize;

child->genom[g1] = par1->genom[g1];
child->genom[g2] = par1->genom[g2];

for (i = 0, j = 0; i < par2->gensize && j <
   par2->gensize; i++, j++) {
        if (j == g1 || j == g2) {
                i--;
                continue;
        }
```

```
                if (par2->genom[i] == child->genom[g1]
                    || par2->genom[i] ==
                    child->genom[g2]) {
                        j--;
                        continue;
                }

                child->genom[j] = par2->genom[i];
        }

        return SUCCESSFUL;
}

int gensequal2(struct Individ *indiv1, struct Individ
    *indiv2, struct Figure *figset)
{
        int i;

        if (indiv1->gensize != indiv2->gensize)
                return 0;

        for (i = 0; i < indiv1->gensize; i++) {
                int a, b;

                a = indiv1->genom[i];
                b = indiv2->genom[i];

                if (figset[a].id != figset[b].id)
                        return INDIVIDS_UNEQUAL;
        }

        return INDIVIDS_EQUAL;
}
```

```
int gensequal(struct Individ *indiv1, struct Individ
  *indiv2)
{
        int i;

        if (indiv1->gensize != indiv2->gensize)
                return 0;

        for (i = 0; i < indiv1->gensize; i++)
                if (indiv1->genom[i] !=
                  indiv2->genom[i])
                        return INDIVIDS_UNEQUAL;

        return INDIVIDS_EQUAL;
}

int mutate(struct Individ *src, struct Individ
  *mutant, int setsize)
{
        int n1, n2, tmp, i;

        if (src->gensize == 1)
                return SMALL_INDIVID;

        mutant->gensize = src->gensize;
        mutant->genom = (int*)xmalloc(sizeof(int*) *
          setsize);

        srand(time(NULL));

        n1 = rand() % (src->gensize - 1);
        n2 = rand() % (src->gensize);
```

```
        while (n1 == n2)
                n2 = rand() % (src->gensize);

        for (i = 0; i < src->gensize; i++)
                mutant->genom[i] = src->genom[i];

        tmp = mutant->genom[n1];
        mutant->genom[n1] = mutant->genom[n2];
        mutant->genom[n2] = tmp;

        return SUCCESSFUL;
}
```

Список литературы

[1] Dyckhoff H. A typology of cutting and packing problems // European Journal of Operational Research. № 44. 150—152 p.

[2] Залгаллер В. А., Канторович Л. В. Рациональный раскрой промышленных материалов. Новосибирск: Наука, 1971.

[3] Никитенков В.Л., Холопов А.В. Задачи линейного программирования и методы их решения. Сыктывкар, 2008. 143 с.

[4] Benell A. J., Olivera F. J. The geometry of nesting problems: A tutorial // European Journal of Operational Research. 2008. № 184. 399—402 p.

[5] MacLeod C. An Introduction to Practical Neural Networks and Genetic Algorithms For Engineers and Scientists. 85 p.

[6] He Y., Liu H. Algorithm for 2D irregular-shaped nesting problem based on the NFP algorithm and lowest gravity-center principle // Journal of Zhejiang University. 2006. № 7. 571 — 574 p.

[7] Панченко Т. В. Генетические Алгоритмs; под ред. Ю. Ю. Тарасевича. Издательский дом «Астраханский университет». 2007. 16 с.