Санкт-Петербургский государственный университет

Кафедра системного программирования

Свирин Евгений Александрович

Реализация алгоритма решения задачи nesting в библиотеке CoreCVS

Курсовая работа

Научный руководитель:
ст. преп. Пименов А. А.

Санкт-Петербург
2020

Оглавление

Введение

Задача раскроя, также известная как nesting или packing problem, является NP-полной задачей оптимизации. Обычно она ставится так: даны некоторые объекты и их нужно разложить по некоторым другим объектам(корзинам). Различия в постановке этих задач заключаются в том, что упаковывают, куда, какие при этом соблюдают условия и какие параметры стремятся улучшить. Эта проблема берет свои корни в истории промышленности, но серьёзное внимание со стороны ученых получила лишь в середине 20-го века.

Задача раскроя впервые сформулирована Канторовичем в 1939 году. На данный момент существует множество научных статей по этой теме, а также реализаций решений таких задач. Успешные и содержательные имплементации чаще всего относятся к промышленным разработкам и поэтому закрыты. Задача раскроя существенна, например, в бумажной, плёночной, сталепрокатной, стекольных промышленностях и металлобработке. Кроме того, методы, используемые в этой задачи, оказываются полезными в других прикладных задачах, в том числе и в компьютерном зрении.

Данная работа добавляет реализацию нескольких алгоритмов решения одной из задач раскроя в библиотеку CoreCVS[1] на языке C++ в IDE QtCreator. Предложенные реализации применимы, например, для расклада объектов для лазерной резки.

1. Постановка задачи

Условие поставленной задачи формулируются следущим образом: контуры в векторном формате, например, сохраненные в SVG, представляют собой несколько выпуклых многоугольников, и на компьютере выбирается прямоугольник – корзина, в которую их нужно поместить, максимизируя оставшуюся верхнюю прямоугольную полосу, содержащую сторону изначальной коробки. Результат сохраняется как SVG файл. Целью данной работы является выбор алгоритмов решения этой задачи и их реализация в библиотеке CoreCVS. Для этого требуется решить следущие задачи:

- изучение подходов к решению задачи раскроя,
- составление нескольких алгоритмов по решению этой задачи, исходя из изученного материала,
- реализация этих алгоритмов в CoreCVS,
- сравнение их между собой и аналогами,
- добавление некоторых инструментов, сопряженных с задачей.

2. Обзор

2.1. Алгоритмика

Ключевое в алгоритме решение задачи nesting – эвристики. Можно выделить две категории эвристик – выборочную и размещения. Выборочная указывает на то, какой объект выбирать на очередной итерации алгоритма, а размещения – куда его помещать. Также есть множество различных эвристик, непопадающие в эти категориик: LowerMassCenter, DJD. Первая указывает, что на некоторых примерах весьма эффективно понизить центр масс у фигур с помощью поворотов. О DJD эвристике подробно расказано в [2], она не эффективна в поставленной задаче.

Bottom-Left placement – эвристика размещения диктующая из всех возможных позиций для фигуры выбирать нижнюю левую.

Среди выборочных, применимых к поставленной задаче есть: First Fit, First Fit Decreasing, First Fit Increasing. Первая заключается в том чтобы помещать объекты в предоставленном порядке, следущая указывает сортировать по площади по убыванию, а последняя по возрастанию.

Для реализаций эвристик полезны метаэвристики, в работе используется метаэвристика NFP, которая позволяет реализовать размещение. NFP 1[3] - no fit polygon, если даны многоугольники, или, в общем случае, фигуры A и B, то no-fit-polygon(AB) – это фигура, которую отчеркивает покрашенная вершина фигуры B при внешнем невращающем касательном обходе фигуры A, Анлогично innerNFP – при внутреннем обходе. NFP позволяет весьма быстро и легко высчитать возможные позиции для фигуры, что продемонстрировано в SVGnest'е [6]. В статье[3] для выпуклых NFP строится за O(n log n) от суммарного количества вершин, реализованный вариант считает NFP за O(n). Также этот метод полезен в компьютерном зрении.

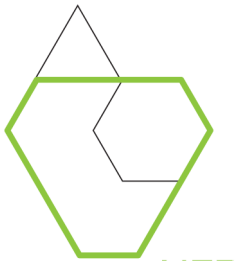

Рис. 1: NFP

2.2. Обзор существующих решений

Есть множество промышленных компаний со своими имплементациями для этой задачи, их код закрыт, да и подробной информации о своих методах они не дают. Это такие компании как: Eastman Machine Company, SIGMATEK, ProNest.

Более близкий аналог моей работы – SVGnest[7], интересен как самый популярный на GitHub и с подходящей теоретической составляющей. Он использует генетический алгоритм, а также поддерживает размещение фигур отличных от многоугольников.

3. Реализация

3.1. Эвристики

Говоря об эвристике размещения, альтернатив Bottom-Left placement, решающих конкретно нашу задачу в изученных статьях не предоставлено. В поиске альтерантивы были изучены примеры работы ProNest, появилось предположение об его методе размещения: задача решается локально на некоторых комбинациях элементов, используя CA эвристику[2], а после из полученных локальных раскладок формирутеся общее решение. При подробном изучении не появилось ясности как рационально организовать этот перебор. В SVGnest, исходя из авторского объяснения и ссылок на статьи, а также изученных примеров его работы используется Bottom-Left Placement. Обычно, пытаясь поместить фигуру, также пытаются поместить несколько её вращений, и из всех позиций выбиратся лучшая. Была придумана новая эвристика размещения, она повышает приоритет позициям, соответствующим фигуре с пониженным центром масс. Следущая формула соответствует новой эвристике.

$$\frac{top(LowerMass(A)_{placed}) - top(A_{placed})}{height(LowerMass(A))} < \sigma$$

В ней top – самая высокая вершина фигуры, height – высота фигуры, LowerMass – метод понижающий центра масс, σ – некоторый параметр, полученный опытным путем. Данное сравнение используется как предикат – помещать ли в позицию с пониженным центром масс. Аналогично, используя заместо высоты фигуры её центр масс, эффективна альтернативная новая эвристика. Стоит отметить, что эвристика заметно улучшает результат при повышении возможного количества поворотов, что видно в 3.3.

Из выборочных эвристик используется FFD, а также было предложено использовать случайную перетасовку для объектов размещения.

3.2. Выбранные алгоритмы

Основываясь на рассуждении 3.1, а также при изучении статей: [2], [3], [4], [5], – было скомпановано несколько алгоритмов решения задачи.

Первый – алгоритм, использующий Bottom-Left placement без поворотов, First Fit Decreasing(FFD) и LowerMassCenter.(рис. 5, 3)

Второй использует новую эвристику и FFD. (рис.6)

Третий использует альтернативную новую эвристику и FFD.

Четвертый использует Bottom-Left placement с поворотами и FFD.

Пятый использует новую эвристику на нескольких случайных перестановках входных данных и выявляет лучшую.

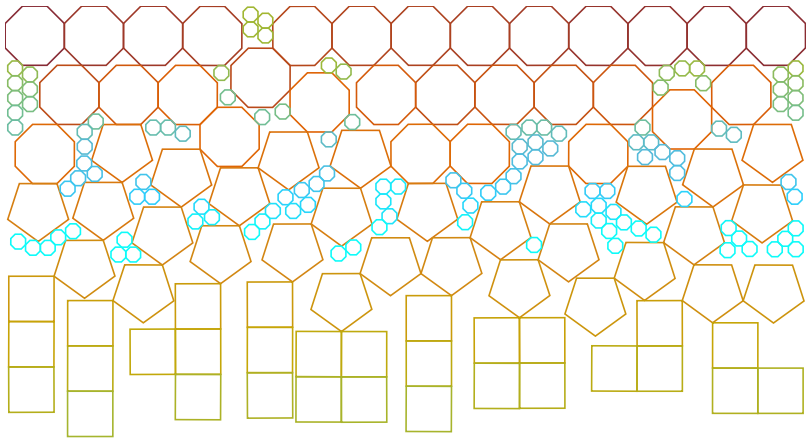

Рис. 2: Раскладка с упорядочиванием по площади.

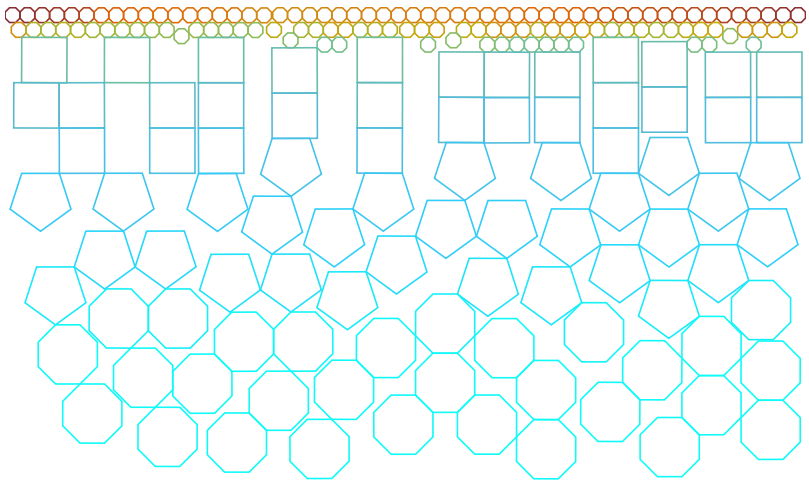

Рис. 3: Раскладка без упорядочивания по площади.

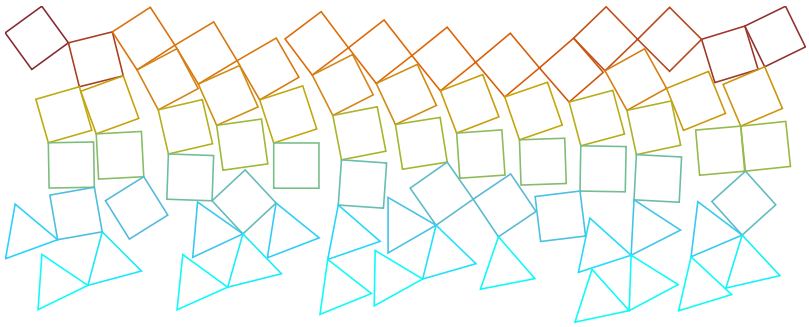

Рис. 4: Многоугольники без понижения центра масс.

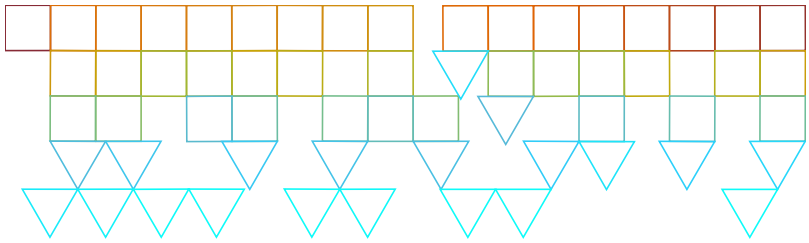

Рис. 5: Они же, только с понижением.

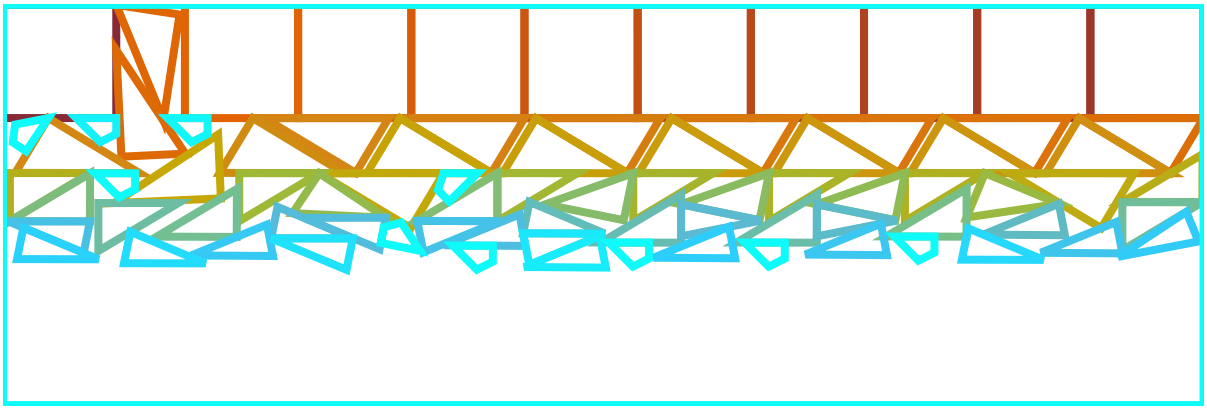

Рис. 6: Пример работы алгоритма с новой эвристикой

Таблица 1: Результат сравненения реализованных алгоритмов

2(4)	2(8)	2(16)	3(4)	3(8)	3(16)	4(16)
0.618	0.612	0.605	0.618	0.620	0.634	0.651
0.952	0.952	0.941	0.961	0.958	0.952	0.945
0.165	0.165	0.162	0.171	0.166	0.172	0.219
0.992	0.992	0.992	0.999	0.999	0.999	0.994

3.3. Сравнение алгоритмов

Тесты составлялись из уже существовавших по этой задаче, дополнялись апроксимациями к промышленным примерам, а также собственными новыми наборами фигур.

Было проведено сравнение реализованных алгоритмов, алгоритмы использующие повороты фигуры проверялись при максимальных допустимых количествах повротов: 4, 8, 16. Следущая таблица – выборка некоторых тестов для самых эффективных алгоритмов из предложенных3.2. Значения в таблице – вертикальная заполненность. Каждая строка соответствует некоторому алгоритму. Вехрняя строка показывает номер алгоритма из 3.2, в скобках указано максимальное допустимое количество поворотов.

3.4. Детали кода

Далее представлен алгоритм получения NFP за линейное время от суммы вершин многоугольников.

A, B - выпуклые многоугольники, A ориентирован по часовой, B - против часовой.

$v1 = getLowLeftVert(A)$
$v2 = getTopRightVert(B)$
$l = size(A) + size(B)$
while l > 0 **do**
  if $angle(v1, next(v1)) < angle(v2, prev(v2))$ **then**
    $add(nfp, v1, next(v1))$
    $v1 = next(v1)$

else

$add(nfp, v2, prev(v2))$

$v2 = prev(v2)$

end if

$l = l - 1$

end while

Пользуясь тем, что сторон выпуклых многоугольников упорядочены по углу с некоторой прямой, производится слияние двух массивов из сторон образуя NFP.

3.5. Сопутствующие методы и инструменты

При использовании готовых инструментов CoreCVS добавлена поддержка форматов DXF, SVG.

С помощью библиотеки Qt реализован простой GUI, использующий самый эффективный алгоритм, и, позволяющий изменять параметры раскладки, поддерживающий форматы DXF и SVG.

Добавлена возможность делать отступы для поддержки резьбы по винилу.

Таблица 2: Результат сравнения с SVGNest

2(32)	2(4)	SVGNest(32)	SVGNest(4)
0.662	0.697	0.674	0.855
0.481	0.481	0.510	0.562
0.461	0.453	0.550	0.581
0.928	0.931	fail	fail

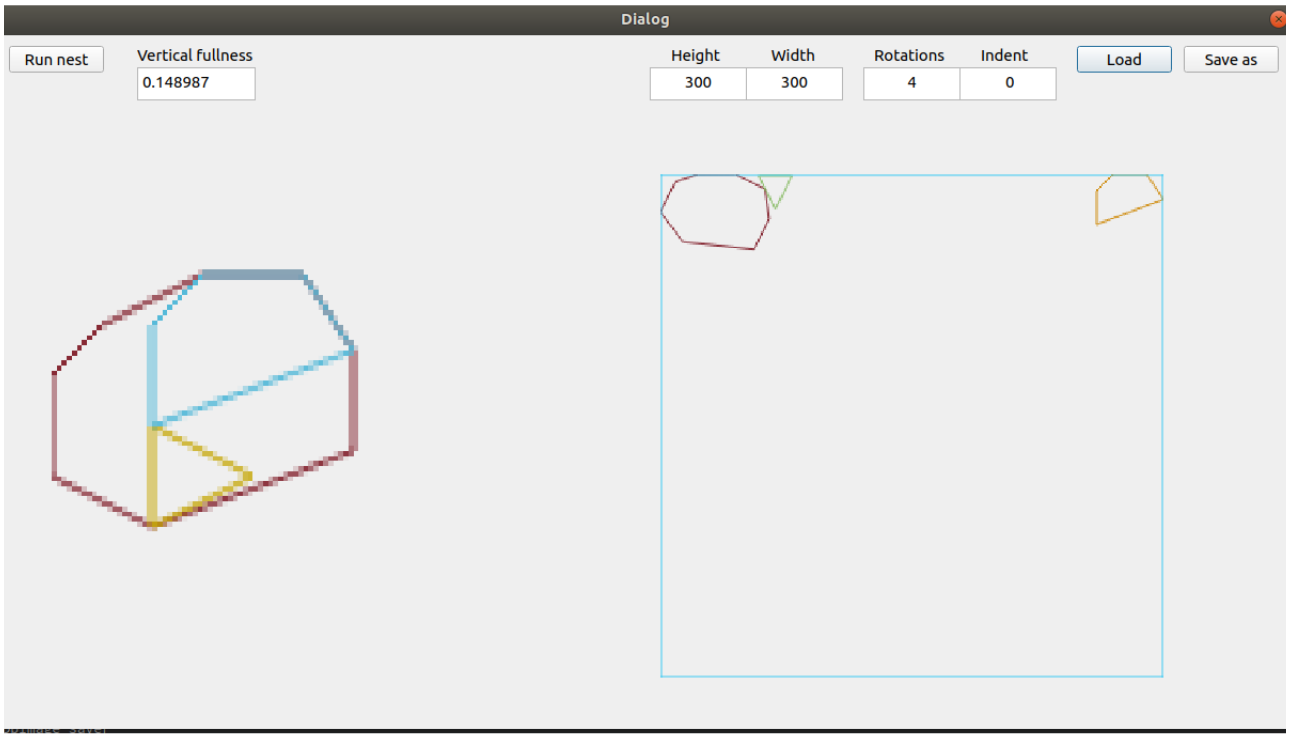

Рис. 7: GUI

3.6. Отладка

Было произведено сравнение лучшего алгоритма с SVGnest, при различных значениях для максимального количества допустимых поворотов. В 2 предоставлена выборка некоторых тестов.

Заключение

Итого, в ходе работы были получены следующие результаты:

- изучен ряд статей по данной теме, придумана и реализована новая эвристика, основывающаяся на повышении приорита позициям соответствующим фигурам с пониженный центрам масс;
- было составлено несколько алгоритмов по решению поставленной задачи;
- в библиотеке corecvs реализовано несколько алгоритмов nesting;
- произведено сравнение реализованных алгоритмов и аналагов;
- добавлена поддержка форматов SVG и DXF, резки по винилу, с помощью инструментов QtCreator реализован простой GUI.

Список литературы

[1] Alexander Pimenov. corecvs. — URL: https://github.com/PimenovAlexander/corecvs.

[2] Burke Eunice López-Camacho · Gabriela Ochoa · Hugo Terashima-Marín · Edmund K. An effective heuristic for the two-dimensional irregular bin packing problem. — URL: http://www.cs.stir.ac.uk/~goc/papers/EffectiveHueristic2DAOR2013.pdf.

[3] E.K. Burke R.S.R. Hellier G. Kendall G. Whitwell *. Complete and robust no-fit polygon generation for the irregular stock cutting problem. — URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.440.379&rep=rep1&type=pdf.

[4] Graham Kendall BSc (Hons). Applying Meta-Heuristic Algorithms to the Nesting Problem Utilising the No Fit Polygon. — URL: http://www.graham-kendall.com/papers/k2001.pdf.

[5] LIU Hu-yao† HE Yuan-jun. Algorithm for 2D irregular-shaped nesting problem based on the NFP algorithm and lowest-gravity-center principle. — URL: http://vmk.ugatu.ac.ru/c%26p/article/A060414.pdf.

[6] Qiao Jack. SVGnest. — URL: https://github.com/Jack000/SVGnest.

[7] Qiao Jack. SVGnest. — URL: https://svgnest.com.