Перевод статьи:
Kokorev D.S. An algorithm for determining the optimal variant of a cut gem with maximal mass and specified symmetry deviations
Business Informatics. 2017. No. 2 (40). P. 40–46.

Алгоритм поиска оптимального варианта огранки драгоценного камня максимальной массы с заданными отклонениями от симметричности[1]

Д.С. Кокорев
аспирант лаборатории распределенных вычислительных систем
Институт проблем передачи информации им. А.А. Харкевича РАН
Адрес: 127051, г. Москва, Большой Каретный пер., д. 19 стр. 1
E-mail: korvin-d@yandex.ru

Аннотация

В статье рассматривается задача нахождения многогранников заданной формы внутри других невыпуклых многогранников. Данная задача является частным случаем третьей части 18-й проблемы Гильберта. Она имеет практическое применение в компьютерном моделировании трехмерных объектов, автономном перемещении роботов, ювелирной промышленности. Эта математическая задача интересна с точки зрения ее применения для нахождения огранок драгоценных камней внутри неограненного камня.

В статье предлагается метод поиска вписанных многогранников, основанный на сведении данной задачи к задаче нелинейного программирования и решения ее с помощью готовых программных средств. Основной идеей является то, что задача легко описывается в терминах нелинейного программирования. Целевой функцией является объем искомого многогранника. Ограничения включают в себя сохранение комбинаторной структуры, содержание многогранника внутри другого, выпуклость и дополнительные ограничения, необходимые для практических целей.

В статье рассмотрены две реализации алгоритма: клиент-серверное приложение и локальное приложение. Приведены их достоинства и недостатки. Алгоритм описан не только с математической точки зрения, но и с точки зрения некоторых его прикладных особенностей. По сравнению с предыдущими статьями автора добавлен метод, который позволяет решать невыпуклый случай задачи, что является значительным шагом с математической точки зрения. Кроме того, это позволяет использовать алгоритм на всех этапах огранки драгоценных камней. В конце статьи даны актуальные оценки эффективности и времени работы алгоритма, в том числе на слабых процессорах, и описаны планы дальнейшего развития алгоритма.

Ключевые слова: выпуклый многогранник, комбинаторная структура, вписанный многогранник, огранка драгоценного камня, отклонение от симметричности, задача нелинейного программирования, солвер.

Цитирование: Kokorev D.S. An algorithm for determining the optimal variant of a cut gem with maximal mass and specified symmetry deviations // Business Informatics. 2017. No. 2 (40). P. 40–46. DOI: 10.17323/1998-0663.2017.2.40.46.

Введение

Одной из классических математических проблем является третья часть 18-ой проблемы Гильберта. В своей общей постановке она посвящается упаковкам одних тел другими. В случае упаковки правильными объектами, например, сферами, существуют доказанные результаты, например, сильная проблема тринадцати сфер [1], проблема двадцати пяти сфер [2] и гипотеза Кеплера [3]. Для случая двухмерных многогранников и простых

[1] Работа выполнена при поддержке Российского научного фонда (проект 16-11-10352)

трехмерных объектов есть ряд решенных задач. Подобные задачи перечислены в статье Ю.И. Валиахметовой и А.С. Филипповой [4]. В случае упаковки многомерных многогранников в другие многогранники задача оказывается очень сложной, и пока не существует теоретических подходов, позволяющих подступиться к этой задаче в общем виде.

Частным случаем этой проблемы является задача нахождения в невыпуклом многограннике произвольной формы объекта заданной формы с наибольшим объемом. Данная задача имеет широкую область применения: компьютерное моделирование трехмерных объектов, программные тренажеры, компьютерные игры, приложения, решающие задачи упаковки и раскроя, программы, отвечающие за передвижение роботов, ювелирная промышленность, обработка дорогостоящих материалов. Описанный в статье алгоритм используется в коммерческих целях для нахождения оптимального плана для огранщика на разных стадиях обработки драгоценного камня.

Первоначальным этапом алгоритма, решающего эту задачу, является нахождение стартовой точки – начального приблизительного расположения объекта. В качестве стартовой точки могут быть взяты результаты работы других алгоритмов. Далее необходимо, перемещая вершины на заданное, не очень большое расстояние, найти положение с максимальным объемом, удовлетворяющее некоторым требованиям к форме.

На данный момент есть несколько программных продуктов для огранщиков, которые определяют план огранки драгоценных камней. Однако все они работают с конечным набором довольно жестко симметричных форм. В качестве стартовой точки для своего алгоритма в статье используется решение этих алгоритмов с искажением их симметрии в допустимых международными и фабричными стандартами рамках, тем самым увеличивая их массу и стоимость.

Алгоритм реализован в клиент-серверном виде в исследовательских целях, а также в локальном виде – в коммерческих целях. Для клиент-серверной реализации алгоритма использованы сторонние программные продукты, организованные с помощью RESTful-веб-сервисов [5]. Такой подход позволяет не тратить время на разработку сложных математических алгоритмов, а использовать уже готовые решения и тем самым сконцентрироваться на поставленной задаче [6]. С терминологией, используемой в статье можно ознакомиться в статье [7].

1. Клиент-серверное приложение

Клиент-серверное приложение состоит из четырех основных этапов работы: обработка входных данных, формирование задачи нелинейного программирования (НП), построение многогранника по полученному решению и вычисления на сервере (*рисунок 1*). Первый этап заключается в преобразовании входных данных, которые могут иметь разные форматы, в необходимый для дальнейшего вычисления формат. На втором этапе исходная геометрическая задача формулируется в терминах нелинейного программирования. После этого на третьем этапе нелинейный солвер решает поставленную НП-задачу. Третий этап осуществляется на сервере и является для пользователя черным ящиком. На самом деле, на нем последовательно запускаются несколько программ, организованных с помощью RESTful-веб-сервисов. После завершения работы солвер выдает набор переменных, определяющих оптимальное решение, по которым на четвертом этапе строится искомый многогранник.

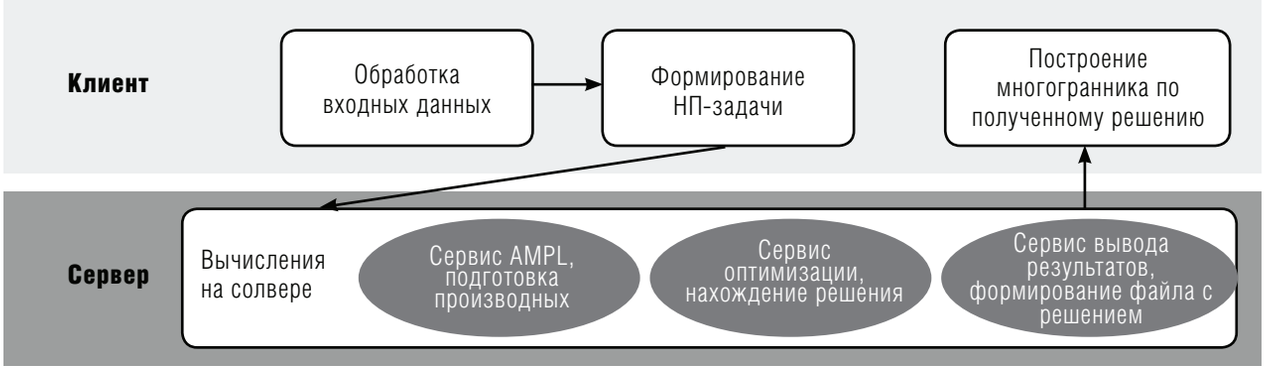

Рис. 1. Схема клиент-серверного приложения

Были протестированы различные солверы для решения задач нелинейного программирования, представленные на сайте neos-server.org. Тесты проводились на задачах поиска оптимального варианта огранки в выпуклых драгоценных камнях. Характеристики задач: около 1000 переменных, около 4000 нелинейных ограничений и 3000 линейных ограничений. Тесты проводились на солверах: CONOPT, Ipopt, Knitro, LANCELOT, LOQO, MINOS, MOSEK, SNOPT. Целевые функции решений отличаются в несущественном знаке. Лучшее время работы на представленном наборе задач показывает бесплатный солвер Ipopt – в среднем 3,50 секунды. Хорошее время работы демонстрируют также платные солверы MINOS и SNOPT – 5,33 и 4,97 секунд соответственно. Также были протестированы солверы глобальной оптимизации на простых задачах вписывания выпуклых Платоновых тел друг в друга. Наилучшие результаты показал солвер Couenne, но он работает в среднем примерно в 300 раз дольше, чем Ipopt, что недопустимо в условиях производства. При этом Ipopt тоже нашел глобальные максимумы на этих простых задачах.

В итоге для вычисления сформулированной НП-задачи в качестве сервиса оптимизации был выбран солвер IPOPT (Internal Point OPTimization) [8]. Этот солвер предназначен для поиска локального оптимума в задаче нелинейного программирования методом внутренней точки. Для вычисления солверу необходимо предоставить callback-функции, вычисляющие следующие величины:

- ✦ значение целевой функции $f(x)$;
- ✦ градиент целевой функции $\nabla f(x)$;
- ✦ вектор значений вектор-функции ограничений $g(x)$;
- ✦ якобиана вектор-функции ограничений $\nabla g(x)^T$;
- ✦ гессиана расширенной функции Лагранжа

$$\sigma_f \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x).$$

Если для формулировки задачи использовать язык программирования AMPL [9], то он сам предоставляет эти функции при получении $f(x)$ и g(x). Таким образом, нужно позаботиться только о формулировке НП-задачи в формате AMPL (формулировка НП-задачи представлена в разделе 4).

Достоинством клиент-серверного подхода является то, что можно одновременно обрабатывать сколько угодно задач при наличии соответствующих ресурсов. Кроме того, лицензия на коммерческое использование AMPL и Ipopt на сервер дешевле, чем лицензии на большое количество локальных приложений.

2. Локальное приложение

Локальное приложение (коробочная версия программы, которая устанавливается на персональный компьютер) нужна, потому что большая часть клиентов программы находится в слаборазвитых странах, где нет стабильного доступа к серверу через Интернет. AMPL является платным и дорогостоящим для коммерческого использования в локальных приложениях, поэтому был разработан и реализован на C++ интерфейс для работы с солвером Ipopt напрямую, без использования AMPL. Были написаны соответствующие callback-функции, упомянутые выше, для узкого множества функций ограничений. Эти функции имеют вид многочленов третьего порядка:

$$\sum c_i x_{i1} x_{i2} x_{i3} + \sum c_j x_{j1} x_{j2} + \sum c_k x_k .$$

Также реализованы функции синуса и косинуса.

Матрицы производных для таких ограничений записываются легко, в отличие от производных для задачи общего вида. Для базового алгоритма без дополнительных ограничений достаточно такого представления для целевой функции и функций ограничений. Более того, большинство геометрических ограничений могут быть аппроксимированы в таком виде с достаточно хорошей точностью.

Целевая функция $f(x)$ и ограничения $g(x)$ хранятся в специальном виде, и Ipopt на каждой итерации обращается к callback-функциям, написанным на C++, которые быстро рассчитывают необходимые производные в текущей точке (*рисунок 2*). Важно минимизировать алгоритмическую сложность вычисления callback-функций, так как эти функции вызываются очень много раз.

Ipopt довольно требователен к процессорному времени. Поэтому одновременно на компьютере можно обсчитывать количество задач равное количеству ядер (удвоенное количество на ядрах с multithreading). Если одновременно поставить больше задач,то результаты будут получаться значительно медленнее. Можно настроить Ipopt так, чтобы одна задача решалась одновременно на нескольких процессорах, но это дает очень маленький коэффициент производительности, что в условиях большого количества маленьких задач не имеет смысла.

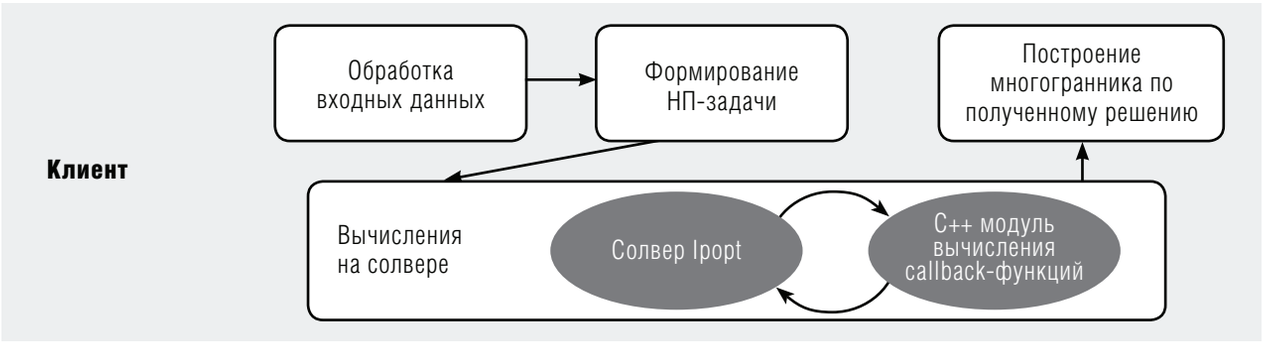

Рис. 2. Схема локального приложения

Для решаемой прикладной задачи важным условием является то, что для любого драгоценного камня за ограниченное время (чаще всего, за минуту) должно быть получено решение. На практике получается, что какие бы фиксированные настройки ни были выбраны, найдутся камни, для которых Ipopt работает дольше отведенного времени. Поэтому все восемь одновременных запусков (если рассматривать стандартный современный четырехъядерный процессор с multithreading) делаются для одного и того же драгоценного камня на разных настройках параметров. Это гарантирует получение результата, а также позволяет получать камни разного качества. Камни разного качества нужны, потому в ювелирной промышленности стоимость ограненного камня зависит не только от массы. Иногда выгоднее сделать более большой камень худшего качества, а иногда камень меньшего размера, но с идеальным качеством.

Кроме того, оказалось, что международные стандарты качества не приспособлены к полноценной работе с несимметричными бриллиантами. У бриллианта, кроме трехмерной симметрии, есть еще и оптическая симметрия – то, насколько хорошо и симметрично он блестит. Приведенные рисунки (*рисунок 3*) отображают ход световых лучей в бриллианте. Чем симметричнее рисунок, тем лучше оптическая симметрия огранки. На *рисунке 3* слева приведена оптическая картинка для идеально ровной огранки массой 1,0149 карат, которая была взята в качестве начальной точки алгоритма. Масса неограненного камня в этом примере составляет 1,2904. Справа приведена оптическая картинка для результата предлагаемого алгоритма, когда включены только ограничения, соответствующие международному стандарту оценки качества бриллиантов. Масса этого решения составляет 1,0601 и формально оно считается решением высшего

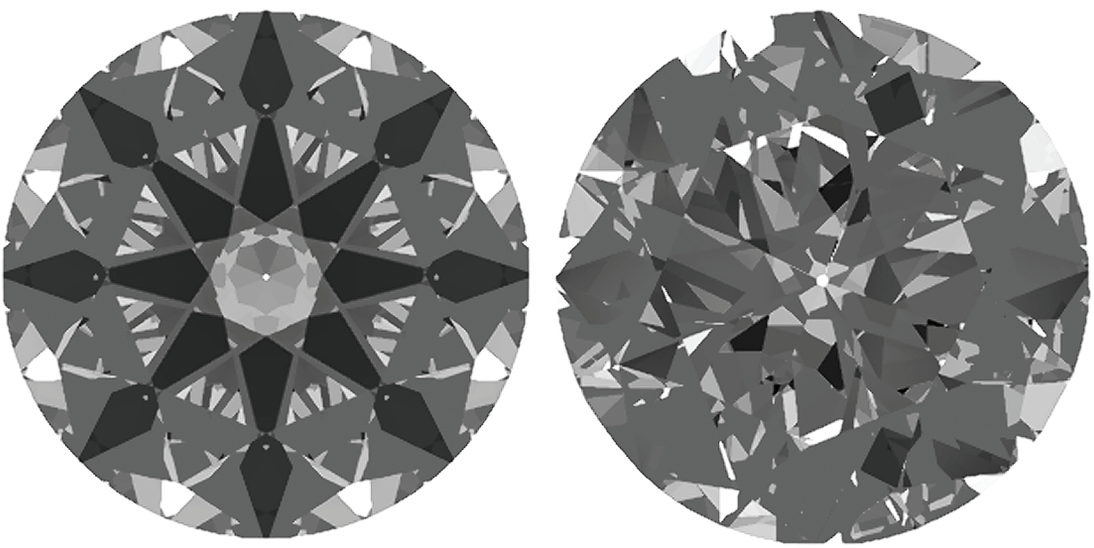

Рис. 3. Оптическая симметрия бриллианта

качества. Но, как можно видеть, оптическая симметрия очень далека от идеальной, а это значит, что бриллиант будет очень хаотично блестеть и его сложно будет продать. Поэтому были изобретены дополнительно параметры, которые контролируют оптическую симметрию бриллианта. Поэтому восемь одновременных запусков настроены так, чтобы получался спектр решений от идеальной оптической симметрии до максимальной массы с допустимым качеством. Для рассмотренного примера получается следующий спектр масс: 1,0601, 1,0428, 1,0406, 1,0368, 1,0356, 1,0316, 1,0315, 1,0268. Все решения, начиная с массы 1,0428, имеют достаточно красивую оптическую симметрию, а решение 1,0268 практически не отличается от идеальной оптической симметрии. Также реализован удобный интерфейс, в котором можно настраивать запуски как угодно, что позволяет клиентам настроить программу под себя и получать огранки, нужного им класса.

3. Задача нелинейного программирования

Поставленную геометрическую задачу необходимо сформулировать в терминах задачи нелинейного программирования.

В качестве целевой функции $f(x)$ в нашем случае берется объем искомой огранки. Он вычисляется через координаты вершин с помощью разбиения граней на треугольники и представления объема в виде суммы объемов симплексов.

Переменные в задаче делятся на два типа. Первый тип – параметры граней огранки y_{ap}, y_{bp}, y_{cp}, y_{dp} , где p – номер грани огранки. В качестве ограничений (xU и xL) на переменные первого типа берутся начальные значения параметров граней, плюс/минус некоторая величина соответственно. Размер этой величины зависит от того, насколько разрешается изменять углы нормалей граней заданной комбинаторной структуры. Параметры граней драгоценного камня не меняются, поэтому рассматриваются как константы. Второй тип переменных – это параметры, отвечающие за расположение вершин огранки в пространстве: $x_{ji} \in (-\infty, +\infty)$, где j – номер точки, i – ось координат. Так как в качестве начальной точки берется решение довольно точного алгоритма, для ускорения работы на координаты вершин можно наложить такие ограничения, чтобы вершины не удалялись от начального состояния дальше, чем на некоторую величину dx.

Перейдем к описанию общих ограничений $g(x)$. Есть несколько основных групп ограничений для данной задачи, без которых нельзя обойтись. Первая группа – ограничения на сохранение комбинаторной структуры огранки:

$$y_{ap}x_{j1} + y_{bp}x_{j2} + y_{cp}x_{j3} - y_{dp} = 0,$$

где p – номер грани огранки; j – номер вершины; индексы 1, 2, 3 соответствуют трем осям координат.

Рассмотренная группа ограничений записывается для всех пар «вершина – грань», для которых верно, что вершина лежит на грани. Перечень всех таких пар получается из начального представления граней многогранника.

Вторая группа – ограничения на выпуклость искомого многогранника. Необходимо записать для всех пар «вершина – грань» огранки, для которых вершина не принадлежит грани:

$$y_{ap}x_{j1} + y_{bp}x_{j2} + y_{cp}x_{j3} - y_{dp} \le 0.$$

Эти ограничения можно записывать только для всех пар соседних граней, потому что попарная выпуклость всех соседних граней является достаточным условием выпуклости всего многогранника.

Последняя группа обязательных условий связана с тем, что огранка должна остаться вписанной в драгоценный камень. Эти условия записываются в следующем виде:

$$A_r x_{j1} + B_r x_{j2} + C_r x_{j3} - D_r \le 0,$$

где r – номер грани драгоценного камня; j – номер вершины огранки.

Следует отметить, что A_r, B_r, C_r, D_r являются константами, а не переменными. Это означает, что ограничения данной группы являются линейными и незначительно влияют на время работы солвера.

Помимо обязательных ограничений, нужно описать все необходимые ограничения на симметрию огранки. Более подробная формулировка НП-задачи для случая, когда драгоценный камень является выпуклым (частично ограненным) представлена в статьях [7, 10]. Перейдем к описанию невыпуклого случая.

Для случая, когда драгоценный камень является невыпуклым, необходимы дополнительные ограничения, отделяющие огранку от невыпуклых частей камня. Сначала строится выпуклая оболочка камня. Для описанных ранее ограничений используются плоскости этой выпуклой оболочки. Далее необходимо вычислить, какие грани

камня являются невыпуклыми. Это делается на основе критерия, в соответствии с которым грань является невыпуклой, если в многограннике есть две вершины, лежащие по разные стороны от этой грани. Далее для каждой такой невыпуклой грани необходимо создать свою разделяющую плоскость. Разделяющая плоскость задается свободными переменными y_a, y_b, y_c, y_d. На эту разделяющую плоскость накладываются ограничения, что все вершины невыпуклой плоскости, привязанной к ней, лежат по одну сторону от нее, а все вершины огранки – по другую сторону. Такие уравнения задаются для каждой невыпуклой грани внешнего многогранника. Если для вершин огранки используется максимальное отклонение от начального положения *dx*, то можно отделять разделяющей плоскостью не все вершины вписываемого многогранники, а только вершины тех граней, которые могут пересечься с невыпуклой гранью внешнего многогранника. Зная значение *dx*, несложно определить набор таких граней для каждой невыпуклой грани внешнего многогранника. Для упрощения работы солвера можно дополнительно задать ограничение, что длина нормали разделяющей плоскости близка к 1.0.

Заключение

Представленный алгоритм реализован в виде клиент-серверного приложения и локального приложения и протестирован на задачах разного размера. Задачи варьировались от простых учебных примеров с простыми, подобными или просто легко вписываемыми многогранниками с количеством вершин меньше пятидесяти до реальных прикладных задач с количеством вершин от ста до нескольких тысяч в невыпуклом случае, и ограничениями на симметрию разной жесткости.

Для тестирования алгоритма была использована специально написанная на языке Python система удаленного тестирования, которая запускает алгоритм на заданной базе камней и собирает отчет содержащий данные по всем запускам и всем контролируемым параметрам. Более подробно система тестирования описана в статье [7].

На реальных примерах алгоритм улучшает любой результат других алгоритмов в исследуемой прикладной области на 1,5–5% объема в зависимости от заданных ограничений на симметрию. Средний прирост массы решений, которые могут быть допущены для огранки драгоценных камней составляет 3%.

Основные тестируемые примеры содержали вписываемый многогранник с количеством вершин и граней порядка 150 и внешний многогранник с количеством вершин и граней порядка 500, общее количество ограничений порядка 10 000. Для примеров с большим количеством невыпуклостей количество ограничений может составлять около 40 000. Время построения задачи на этих примерах составляет около одной секунды, время вычисления на солвере без ограничений на симметрию – около 5 секунд, время вычислений на солвере со стандартными ограничениями на симметрию – в среднем 20 секунд. Для невыпуклых камней среднее время построения задачи составляет 29 секунд. Данные приведены для процессора Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz. Также были проведены тесты с заниженной частотой процессора для имитации более слабых компьютеров. На максимальных частотах 0,9, 1,4, 1,9, 2,4, 2,9 GHz среднее время работы с ограничениями составляет 90, 57, 42, 34, 28 секунд соответственно.

В статье рассмотрены две реализации алгоритма: с использованием удаленного вычислительного ресурса и без. Приведены их достоинства и недостатки. Описан сам алгоритм не только с математической точки зрения, но и с точки зрения некоторых его прикладных деталей и нюансов. По сравнению с предыдущей статьей [7] найден метод, который позволяет решать невыпуклый случай задачи, что является значительным шагом с математической точки зрения, а также позволяет использовать алгоритм на всех этапах огранки драгоценных камней. Даны актуальные оценки эффективности и времени работы алгоритма, в том числе на слабых процессорах. На данном этапе алгоритм реализован для наиболее распространенной круглой огранки. В дальнейшем планируется обобщить его и для других огранок. ■

Литература

1. Tarasov A., Musin O. The strong thirteen spheres problem // Topology, Geometry, and Dynamics: Rokhlin Memorial. Saint Petersburg, Russia. 11–16 January 2010.
2. Мусин О.Р. Проблема двадцати пяти сфер // Успехи математических наук. 2003. Т. 58, № 4 (352). С. 153–154.
3. Hales T. The status of the Kepler conjecture // Mathematical Intelligencer. 1994. No. 16. P. 47–58.

4. Валиахметова Ю.И., Филиппова А.С. Теория оптимального использования ресурсов Л.В. Канторовича в задачах раскроя – упаковки: Обзор и история развития методов решения // Вестник УГАТУ. 2014. Т. 18, № 1 (62). С. 186–197.
5. Афанасьев А.П., Волошинов В.В., Лисов А.А., Наумцева А.К. Организация распределенной обработки данных с помощью RESTful-веб-сервисов // Современные проблемы науки и образования. 2012. № 6 (приложение «Технические науки»). С. 31.
6. Волошинов В.В., Смирнов С.А. Принципы интеграции программных ресурсов в научных вычислениях // Информационные технологии и вычислительные системы. 2012. № 3. С. 66–71.
7. Кокорев Д.С. Оптимизационный алгоритм поиска вписанного многогранника максимального объема // Программные продукты и системы. 2016. № 1. С. 90–95.
8. Kawajir Y., Laird C., Wächter A. Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt. [Электронный ресурс]: http://www.coin-or.org/Ipopt/documentation/ (дата обращения 15.05.2017).
9. Fourer R., Gay D.M., Kernighan B.W. AMPL: A modeling language for mathematical programming. Belmont: Duxburry Press, 2003.
10. Кокорев Д.С. Алгоритм поиска выпуклого многогранника максимального объема, вписанного в другой многогранник // Информационные технологии и вычислительные системы. 2013. №3. С. 27–31.

Перевод статьи:
Kokorev D.S. An algorithm for determining the optimal faceting variant of a gem with maximal mass and specified symmetry deviations
Business Informatics. 2017. No. 2 (40). P. 40–46.