УДК 004.93’11 + 519.173 + 515.124.4
DOI: 10.25686/978-5-8158-2474-4-2025-878-898

Извлечение геометрических объектов из пространства чертежа CAD-модели посредством изоморфизма графов

И. А. Федотов[1,2]
[1]Navio, Москва, Россия
[2]Нижегородский государственный университет им. Н.И. Лобачевского, Нижний Новгород, Россия

Аннотация. Работа направлена на решение проблемы логического распознавания объектов и их образов посредством задач изоморфизма графа совместно с дискретной метрической геометрией. В качестве постановки выбрана задача нахождения известной геометрической формы из словаря, в пространстве CAD модели, где известны координаты точек и связи между ними. От более общей задачи распознавания по проекции трехмерного объекта на сетчатку (ПЗС-матрицу) постановка отличается тем, что нет проекционных искажений, связи представляют собой прямые без функционального пространства кривых и параметрических поверхностей. Уровни детализации объекта опущены ввиду пока излишней сложности задачи гомеоморфизма подграфа. Допущение на знание координат не принципиально, достаточно расстояний между точками. С учетом допущений можно трактовать постановку как распознавание образов геометрических объектов по изометрической проекции на сетчатке.

С практической точки зрения предложенный метод позволяет решать новые классы задач: поиск конкретной или подобной детали в сборке, поиск максимально близкой детали или общей части/границы, версионирование CAD-модели (выделение максимально общей части), поиск разных классов симметрий, анализ деформации объекта. Обеспечивает унификацию процесса распознавания, сводя разные классы задач к типовой задаче сопоставления графов, исключительно в машинной обработке. Например, определение предмета и его ориентации относительно наблюдателя по силуэту или его части - как задачу поиска атрибутированного по ребрам цикла/цепи через поиск неиндуцированного изоморфного подграфа или его максимального вхождения.

***Ключевые слова*:** изоморфизм графов, сопоставления графов, дискретная метрическая геометрия, геометрические формы, распознавание образов, системы компьютерного проектирования.

Geometric forms extraction from CAD model space through graph isomorphism approaches

I. A. Fedotov[1,2]
[1]Navio, Moscow, Russia
[2]Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia

Abstract. This work aims to solve the problem of logical recognition of objects and their patterns using graph isomorphism problems combined with discrete metric geometry. The chosen problem setting is finding a known geometric shape from a dictionary in the model space of a CAD model, where the coordinates of points and the relationships between them are known. This setting differs from the more general problem of recognition based on the projection of a three-dimensional object onto the retina (CCD matrix) in that there are no projection distortions; the relationships are straight lines, without the functional space of curves and parametric surfaces. Levels of object detail are omitted due to the currently excessive complexity of the subgraph homeomorphism problem. The assumption of knowing the coordinates is not essential; the distances between points are sufficient. Given these assumptions, the setting can be interpreted as pattern recognition of geometric objects based on an isometric projection onto the retina.

From a practical perspective, the proposed method enables solving new classes of problems: searching for a specific or similar part in an assembly, searching for the closest part or common part/boundary, versioning a CAD model (identifying the most common part), searching for different symmetry classes, and analyzing object deformation. It unifies the recognition process, reducing various classes of problems to a typical graph matching task, exclusively for machine processing. For example, determining an object and its orientation relative to the observer based on its silhouette or part of it is like searching for an edge-attributed cycle/chain by searching for a non-induced isomorphic subgraph or its maximum common subgraph.

***Keywords*:** graph isomorphism, graph matching, distance discrete geometry, geometry forms, pattern recognition, CAD.

Введение

They drew all manner of things— everything that begins with an M—.

Lewis Carroll Alice's Adventures in Wonderland. United Kingdom *1865* like **M**atching, **M**ορφη΄, and the Cayley-**M**enger determinant - you know, you say things are 'a lot alike' – have you ever seen such a thing as a graph similarity drawing?

Геометрическія фигуры. Всякая ограниченная часть пространства называется геометрическимъ теломъ.
Геометрическое тело можно подразделять на части; каждая часть геометрического тела есть также геометрическое тело.
Граница геометрическаго тела, т.е. то, чемъ оно отделяется отъ остального пространства, наз. поверхностью.
Граница поверхности называется линіей.
Граница линіи называется точкой.

А. Киселевъ
Элементарная геометрія. Москва 1914 г.

Используем определение геометрической фигуры Киселева, вынесенное во второй эпиграф для отображения геометрических понятий на теорию графов и дискретную геометрию. Так, поверхность (грань) будем отождествлять с гиперребром гиперграфа, линию – с ребром графа, а точку – с узлом. Такое обобщение достаточно мощно и может в будущем использоваться для сложных поверхностей и кривых, в нашей статье рассматриваются связи в виде линий, и приставка гипер не требуется, разве что для описания граней.

- Геом. тело/фигура (совокупность граней/поверхностей) → Гиперграф, $H(V, E, L)$
- Поверхность/грань → Гиперребро, E
- Линия/прямая (пересечение поверхностей) → Ребро, $e \in E(H)$
- Параметры линии (длина, кривизна, ...) → Атрибуты ребер мультиграфа, L_i^e
- Точка (пересечение линий) → Узел, $v \in V(H)$
- Параметры точки (координаты, тип, ...) → Атрибуты узла, L_i^v

При описании движения и эволюции графа в будущем, возможно, потребуется привлечь время. Графы, которые эволюционируют во времени, называют темпоральные (temporal) или time-evolving, общая запись для них: $H(V(t), E(t), L(t))$, т.е. эволюционировать могут вершины, ребра и метки независимо друг от друга. Некоторые авторы такой вид графов сводят к мультиграфам, считая время дискретным и добавляя новую размерность к меткам, тем самым вводя тензора для общности с машинным обучением. Есть другой путь: оставить время аналоговым (дискретным), а эволюцию множеств $V(t)$, $E(t)$, $L(t)$ выразить в виде дифференциальных (разностных) уравнений, тем самым соединив дискретную математику и аналитическую механику, если мы говорим про динамические системы.

Пока умозрительно видится, что геометрическую фигуру можно описать методами теории графов (дискретной геометрии), сложности возникают в технике описания: как точно описать форму фигуры, чем можно пренебречь, а что действительно необходимо для последующего распознавания? Далее представим своё видение проблемы и способы её решения на примере нескольких практических задач.

Отдельно стоит сказать о графовых алгоритмах на изоморфизм (табл. 1), они не являются основным предметом статьи, хотя и играют ключевую роль, потенциально способные свести на нет все потуги теоретических соображений. Действительно, задачи являются NP полными, с волнующим факториалом в оценке производительности, но на сегодняшний день существуют эффективные алгоритмы, способные работать в реальном времени (за исключением пока разве что алгоритма SHOM и частично MCS) на больших плотных графах. Автор как раз является разработчиком нескольких из них, в том числе и специально спроектированных для геометрических задач, и поэтому дальнейшее повествование пойдет в предположении, что проблема с алгоритмами не является краеугольной. Она может быть эффективно решена даже на относительно скромных по нынешним меркам системах уровня desktop и даже на embedded уровне. Ссылки и некие обзоры алгоритмов можно найти в библиографии работ [18, 19] или на странице автора.

В статье для простоты, а также будущего перехода к проекциям на сетчатке, сосредоточимся на плоских фигурах, состоящих из четырех точек, называемых 4-угольниками, и попытаемся представить их описание, опираясь исключительно на графовое представление. Покажем, что с помощью понятия изоморфизмов и введенной матрицы формы можно находить объекты на чертеже после аффинных и общих (даже разрушающих) преобразований. Помимо этого с помощью расстояний (мировой функции [4–7]) и понятий из дискретной геометрии [1–3] в виде длин отрезков и полного графа из них, станет возможен переход в инвариантное пространство, не зависящее ни от координат (положения в пространстве), ни от реальных размеров предмета. Такой переход в будущем позволит создавать антропоцентрическую систему координат (систему, создаваемую субъектом познания, машиной или человеком) по ходу процесса познания окружающего мира, что является главным моментом в распознавании и отождествлении образов.

Текст пронизан сквозным примером четырехугольника типа kite, этот тип относительно прост (среди всех типов Quadrilaterals), но уже сложнее квадрата и ромба.

Таблица 1. **Задачи сопоставления графов (graph matching)**

Task name	Canonical name	Description
MCP	Maximal Clique Problem	Поиск полного подграфа, разложение на независимые клики. Структурный анализ графа, выделение основных структурных компонент и их взаимосвязей
AUT	Automorphism	Поиск автоморфизма графа. Разложение на симметрии графа, только структурные или с учетом семантики
GI	Graph Isomorphism	Сравнение графов на изоморфность, эквивалентность форм
SI	Subgraph Isomorphism	Нахождение подграфов изоморфных образцу из сложного графа чертежа
MCS	Maximum Common Subgraph	Нахождение максимально общей изоморфной части двух графов (пересечение графов)
SHOM	Subgraph Homeomorphism	Выделение подграфов или их максимально общей изоморфной части с учетом уровня детализации (абстракции) на большом чертеже

Задача дискретной геометрии: Дельтоид (Ромбоид) / Kite

Представим kite в безразмерном параметрическом виде с помощью соотношений диагоналей $\frac{|bc|}{|ad|} = \beta = \frac{2}{4} = 0.5$ и разбиением большой диагонали $\frac{|od|}{|ao|} = \gamma = \frac{1}{3}$.

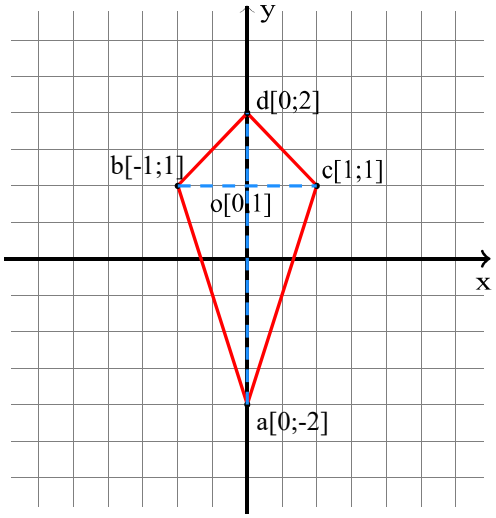

Рисунок 1. Kite

Это не единственный способ представления, но он нам будет удобен в последующем, хотя на самом деле главное, что заданы все координаты точек, а параметрический вид нужен для аналитического описания фигуры. Т.е., расположив четыре точки на плоскости, мы однозначно задаём четырех угольник, его форму определяют опять же координаты точек, т.е. их взаимное расположение. С точки зрения теории графов нашу фигуру можно описать неориентированным графом с четырьмя вершинами и четырьмя ребрами (матрицей смежности) и весовой матрицей. Некоторые авторы представляют такой граф исключительно весовой матрицей, но мы для удобства будем выделять структурную компоненту – булеву матрицу смежности **A** и семантическую – вещественную матрицу весов **W**. Тогда

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}. \tag{1}$$

Поскольку граф не ориентирован ($A = A^T$) и без петель ($diag(A) = 0$), нам для однозначного задания достаточно только верхнего надтреугольника. Вместе с тем наряду с матрицей **A** незримо присутствует матрица $\overline{A} - I$, где $\overline{A}$ – логическое отрицание, I – единичная матрица (нужна для устранения несуществующих петель, которые неизбежно появляются при инверсии), для краткости такую матрицу можно также обозначать как $\overline{A}$, подразумевая, что петли задачей не предусмотрены, такая матрица называется дополнением графа и всегда незримо существует, т.е. имманентна основному графу.

$$\overline{A} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Видно, что при принятой индексации вершин диагонали ромбоида $|ad|, |bc|$ появляются на антидиагонали матрицы $\overline{A}$.

Описанное выше элементарно, некоторый интерес представляет весовая матрица, её представим как полную матрицу длин всех отрезков (существующих и несуществующих ребер) и нормированную на максимальное значение, пропустив (оставив пытливым читателям) упражнения по математической геометрии, по Рылову [8], имеем

$$\mathbf{M} = \begin{pmatrix} 0 & \sqrt{\frac{1}{(1+\gamma)^2} + \frac{\beta^2}{4}} & \sqrt{\frac{1}{(1+\gamma)^2} + \frac{\beta^2}{4}} & 1 \\ \sqrt{\frac{1}{(1+\gamma)^2} + \frac{\beta^2}{4}} & 0 & \beta & \sqrt{\frac{\gamma^2}{(1+\gamma)^2} + \frac{\beta^2}{4}} \\ \sqrt{\frac{1}{(1+\gamma)^2} + \frac{\beta^2}{4}} & \beta & 0 & \sqrt{\frac{\gamma^2}{(1+\gamma)^2} + \frac{\beta^2}{4}} \\ 1 & \sqrt{\frac{\gamma^2}{(1+\gamma)^2} + \frac{\beta^2}{4}} & \sqrt{\frac{\gamma^2}{(1+\gamma)^2} + \frac{\beta^2}{4}} & 0 \end{pmatrix}, \tag{2}$$

где параметры β, γ определены в начале раздела. Видно, что матрица у нас симметрическая, положительная, с элементами, не превышающими 1. На главной диагонали стоят нули, что в вычислительном плане не очень хорошо, в дальнейшем главную диагональ можно заполнить весами

узлов, метками или другими атрибутами. На самом деле мы получили гораздо больше, чем матрицу весов, полученная нами матрица имеет физический смысл – формы фигуры, она задана в отношениях, т.е. независима относительно исходных координат и положения фигуры в пространстве, и её абсолютных размеров. Сам её вид специфичен, т.е. заполнение числами уникально для каждой фигуры, поэтому переобозначим её через $\mathbf{M}$ от греческого слова *Μορφη΄* (форма). На самом деле $\mathbf{M}$ есть ни что иное, как нормированный (обезразмеренный) (1,1)-минор определителя Кэли-Менгера [1–3].

Тогда существующие ребра (длины сторон) извлекаются как

$$M = A \circ \mathbf{M},$$

а несуществующие (диагонали) как

$$\overline{M} = \overline{A} \circ \mathbf{M},$$

где $\circ$ – покомпонентное матричное произведение (Адамара, Шура) [13].

Для всех четырехугольников матрица $\mathbf{A}$ будет неизменна, меняется только матрица формы $\mathbf{M}$. Что, собственно говоря, логично, поскольку в математическом плане это всего лишь изоморфизмы некого образа. В вычислительном плане исходная задача изоморфизма дополняется матрицей формы и на языке задач сопоставления графов называется как изоморфизм атрибутированного по ребрам подграфа.

Теперь представим численное задание матрицы формы, поскольку очевидно, что аналитическое достаточно трудозатратно, благо можно получить матрицу $\mathbf{M}$, используя только координаты вершин.

$$a=(0.0;-2.0);\quad b=(-1.0;1.0);\quad c=(1.0;1.0);\quad d=(0.0;2.0);$$

$$\begin{aligned}
e_{ab} &= \sqrt{(x_b-x_a)^2+(y_b-y_a)^2} = \sqrt{(-1-0)^2+(1+2)^2} = \sqrt{10} \approx 3.1623;\\
e_{bd} &= \sqrt{(x_d-x_b)^2+(y_d-y_b)^2} = \sqrt{(0+1)^2+(2-1)^2} = \sqrt{2} \approx 1.4142;\\
e_{dc} &= \sqrt{(x_c-x_d)^2+(y_c-y_d)^2} = \sqrt{(1-0)^2+(1-2)^2} = \sqrt{2} \approx 1.4142;\\
e_{ca} &= \sqrt{(x_a-x_c)^2+(y_a-y_c)^2} = \sqrt{(0-1)^2+(-2-1)^2} = \sqrt{10} \approx 3.1623;\\
grayblacke_{ad} &= \sqrt{(x_d-x_a)^2+(y_d-y_a)^2} = \sqrt{(0-0)^2+(2+2)^2} = \sqrt{4^2} = 4;\\
e_{bc} &= \sqrt{(x_c-x_b)^2+(y_c-y_b)^2} = \sqrt{(1+1)^2+(1-1)^2} = \sqrt{2^2} = 2.
\end{aligned} \tag{3}$$

Найдем максимальное ребро: очевидно, это неявное ребро - диагональ: $e^{\text{e}} = e_{ad} = e_{bc} = 4$, тогда матрица формы

$$\mathbf{M} = \begin{pmatrix} & a & b & c & d \\ a & 0 & \frac{e_{ab}}{e^{\text{e}}} & \frac{e_{ac}}{e^{\text{e}}} & \frac{e_{ad}}{e^{\text{e}}} \\ b & \frac{e_{ba}}{e^{\text{e}}} & 0 & \frac{e_{bc}}{e^{\text{e}}} & \frac{e_{bd}}{e^{\text{e}}} \\ c & \frac{e_{ca}}{e^{\text{e}}} & \frac{e_{cb}}{e^{\text{e}}} & 0 & \frac{e_{cd}}{e^{\text{e}}} \\ d & \frac{e_{da}}{e^{\text{e}}} & \frac{e_{db}}{e^{\text{e}}} & \frac{e_{dc}}{e^{\text{e}}} & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{\sqrt{10}}{e^{\text{e}}} & \frac{\sqrt{10}}{e^{\text{e}}} & \frac{4}{e^{\text{e}}} \\ \frac{\sqrt{10}}{e^{\text{e}}} & 0 & \frac{2}{e^{\text{e}}} & \frac{\sqrt{2}}{e^{\text{e}}} \\ \frac{\sqrt{10}}{e^{\text{e}}} & \frac{2}{e^{\text{e}}} & 0 & \frac{\sqrt{2}}{e^{\text{e}}} \\ \frac{4}{e^{\text{e}}} & \frac{\sqrt{2}}{e^{\text{e}}} & \frac{\sqrt{2}}{e^{\text{e}}} & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{\sqrt{5}}{2\sqrt{2}} & \frac{\sqrt{5}}{2\sqrt{2}} & 1 \\ \frac{\sqrt{5}}{2\sqrt{2}} & 0 & \frac{1}{2} & \frac{1}{2\sqrt{2}} \\ \frac{\sqrt{5}}{2\sqrt{2}} & \frac{1}{2} & 0 & \frac{1}{2\sqrt{2}} \\ 1 & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & 0 \end{pmatrix}.$$

При произвольной индексации вершин матрица будет иметь другой вид, но значения будут те же, только стоять они будут в других местах. На языке теории графов это называется изоморфизм, т.е. сохраняющий/независимый от формы.

Матрица смежности $\mathbf{A}$ задаётся видимыми ребрами, например, через команду polyline (autocad) для любой индексации вершин

$$polyline\ [0;2],[1;1],[0;-2],[-1;1],[0;2];\quad //d \to c \to a \to b \to d.$$

1. Построение матрицы формы $\mathbf{M}$

Приведем конструктивную процедуру построения матрицы формы: пусть у нас имеются множество точек фигуры, для нашего примера это *a*,*b*,*c*,*d*, сформируем из него матрицу точек G

$$G = \begin{pmatrix} a & b & c & d \end{pmatrix} = \begin{pmatrix} x_a & x_b & x_c & x_d \\ y_a & y_b & y_c & y_d \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & 0 \\ -2 & 1 & 1 & 2 \end{pmatrix}. \tag{4}$$

Мы записали матрицу G в привычном представлении, где вектора точек располагаются по столбцам, при обработке, однако, удобнее иметь транспонированное представление – G^T, поскольку согласно Fortran нотации *x* координаты будут лежать в одном столбце, *y* – в другом и т.д., при таком хранении увеличивается скорость работы за счет равномерного доступа к данным и векторизации, в нашем случае по столбцам.

Из матрицы (4) точек получим матрицу векторов (6) (полный граф межточечных соединений), но поскольку это будет уже многомерная матрица (тензор), для привычного двумерного вида разобьём её на разные компоненты (одну – с координатами *x*, другую – с *y* и т.д.). Тогда

$$\begin{aligned} G_x &= \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot G; \\ G_y &= \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot G. \end{aligned} \tag{5}$$

$$\begin{aligned} V_x &= j \otimes G_x - G_x^T \otimes j^T; \\ V_y &= j \otimes G_y - G_y^T \otimes j^T, \end{aligned} \tag{6}$$

где $\otimes$ – произведение Кронекера, а *j* – вектор-столбец, состоящий из единиц (столбец матрицы J) $j^T = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$

$$V = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} a & b & c & d \end{pmatrix} - \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} a-a & b-a & c-a & d-a \\ a-b & b-b & c-b & d-b \\ a-c & b-c & c-c & d-c \\ a-d & b-d & c-d & d-d \end{pmatrix};$$

$$V = \begin{pmatrix} \overrightarrow{aa} & \overrightarrow{ab} & \overrightarrow{ac} & \overrightarrow{ad} \\ \overrightarrow{ba} & \overrightarrow{bb} & \overrightarrow{bc} & \overrightarrow{bd} \\ \overrightarrow{ca} & \overrightarrow{cb} & \overrightarrow{cc} & \overrightarrow{cd} \\ \overrightarrow{da} & \overrightarrow{db} & \overrightarrow{dc} & \overrightarrow{dd} \end{pmatrix} = \begin{pmatrix} 0 & \overrightarrow{ab} & \overrightarrow{ac} & \overrightarrow{ad} \\ \overrightarrow{ba} & 0 & \overrightarrow{bc} & \overrightarrow{bd} \\ \overrightarrow{ca} & \overrightarrow{cb} & 0 & \overrightarrow{cd} \\ \overrightarrow{da} & \overrightarrow{db} & \overrightarrow{dc} & 0 \end{pmatrix},$$

где $\vec{a}b = \begin{bmatrix} x_b - x_a \\ y_b - y_a \end{bmatrix}$, например. Тогда матрица длин векторов будет очевидно симметрична $L = L^T$ и иметь вид

$$L = \sqrt{\sum_{i=1}^{dim} \left(V_i \circ V_i \right)}, \tag{7}$$

где $\sqrt{\ }$ – поэлементный квадратный корень (matlab: sqrt()/realsqrt(), julia: sqrt.()) или в записи через нормы

$$L = \begin{pmatrix} 0 & \mathrm{P}\vec{a}b\,\mathrm{P}_2 & \mathrm{P}\vec{a}c\,\mathrm{P}_2 & \mathrm{P}\vec{a}d\,\mathrm{P}_2 \\ \mathrm{P}\vec{b}a\,\mathrm{P}_2 & 0 & \mathrm{P}\vec{b}c\,\mathrm{P}_2 & \mathrm{P}\vec{b}d\,\mathrm{P}_2 \\ \mathrm{P}\vec{c}a\,\mathrm{P}_2 & \mathrm{P}\vec{c}b\,\mathrm{P}_2 & 0 & \mathrm{P}\vec{c}d\,\mathrm{P}_2 \\ \mathrm{P}\vec{d}a\,\mathrm{P}_2 & \mathrm{P}\vec{d}b\,\mathrm{P}_2 & \mathrm{P}\vec{d}c\,\mathrm{P}_2 & 0 \end{pmatrix}, \tag{8}$$

и матрица формы уже выражается как

$$\mathbf{M} = \frac{L}{\mathrm{P}vec(L)\,\mathrm{P}_\infty} = \frac{1}{\mathrm{P}vec(L)\,\mathrm{P}_\infty} I \cdot L = \frac{1}{k^*} I \cdot L. \tag{9}$$

Из (9) нам нужен только верхний треугольник, и для его получения не обязательно выполнять все матричные операции, достаточно только тех что его определяют. Для этого можно сразу сделать V в нужном виде:

$$vech(V)^T = \begin{pmatrix} \vec{a}b \\ \vec{a}c \\ \vec{a}d \\ \vec{b}c \\ \vec{b}d \\ \vec{c}d \end{pmatrix} = \left(G \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \right)^T - \left(G \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right)^T, \qquad (10)$$

а L рассчитывать по той же формуле (7) уже для матрицы $vech(V)$.

Хотя мы сознательно избавились от углов в задании фигуры, они всё равно присутствуют неявно через теорему косинусов $\cos\gamma = \frac{a^2+b^2-c^2}{2ab}$ [c2], равно как объём и площадь фигуры можно вычислить по матрице длин (8) через формулу Герона: $S = \frac{1}{4}\sqrt{4a^2b^2-(a^2+b^2-c^2)^2}$, триангулируя предварительно фигуру на элементарные треугольники, что собственно и было сделано Артуром Кэли совместно с Карлом Менгером. Совокупность углов будет являться обобщенными координатами (понятие из аналитической механики), наравне с длинами и координатами точек и использовать их можно в тех приложениях, где углы вычислить проще, чем расстояния (например, в сферической астрономии).

В силу специфики статьи для CAD-систем мы ориентировались на естественный в данной области способ задания фигуры посредством координат вершин, но он не единственный. Из записи (9) можно видеть, что нам достаточно знать только длины отрезков, вообще говоря, всех явных и неявных. Можно обойтись только явными, а неявные выразить через них, этот процесс тут опустим, поскольку самое главное, на чем надо акцентировать внимание, – это то, что нам нужны только длины ребер (без углов и координат, и даже не сами длины, а их соотношения), а значит данную технику можно использовать и в сфере реального распознавания образов, т.е. посредством камер с реального мира. Ребра и длины можно выделить и посчитать из данных с ПЗС-матрицы или (для краткости) сетчатки. Постановка задачи там гораздо сложнее, поскольку имеются искажения и вообще нужно уметь распознавать движения и выделять инерциальные системы координат.

2. Преобразования фигуры

При проведении преобразований над точками, матрица длин (7) изменится. Так, вектор по двум точкам (допустим, *a*,*b*) будет

$$\vec{a}b = \begin{bmatrix} x_b - x_a \\ y_b - y_a \end{bmatrix} = \begin{bmatrix} x_{ab} \\ y_{ab} \end{bmatrix}.$$

Его длину можно выразить через внутреннее произведение векторов

$$< \vec{a}b, \vec{a}b >= \vec{a}b^T \cdot \vec{a}b;$$

$$\mathrm{P}\,\vec{a}b\,\mathrm{P}_2 = \sqrt{< \vec{a}b, \vec{a}b >} = \rho = \sqrt{2\sigma},$$

где ρ – расстояние; σ – мировая функция, введенная Рыловым и независимо Сингом [10]. Тогда если мы произведём линейное (или нет) преобразование над точками $a' = Pa, b' = Pb$, где $P = \begin{pmatrix} p_1 & p_2 \\ p_3 & p_4 \end{pmatrix}$, то

$$\vec{a}b' = \begin{bmatrix} p_1(x_b - x_a) + p_2(y_b - y_a) \\ p_3(x_b - x_a) + p_4(y_b - y_a) \end{bmatrix} = \begin{pmatrix} p_1 & p_2 \\ p_3 & p_4 \end{pmatrix} \cdot \begin{bmatrix} x_b - x_a \\ y_b - y_a \end{bmatrix} = P \cdot \vec{a}b,$$

поэтому внутреннее произведение преобразованного вектора будет

$$< \vec{a}b', \vec{a}b' >= \vec{a}b^T P^T \cdot P\vec{a}b.$$

При ортогональных преобразованиях $P = Q$, где $Q^T = Q^{-1}$, произведение $Q^T \cdot Q = I$ и соответственно матрица длин остаётся без изменений, при деформирующих преобразованиях это в общем виде не так. Хотя матрица L не сохраняется при деформирующем равномерном масштабировании $P_s = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$

$$L_s = k\sqrt{\left(V_x \circ V_x + V_y \circ V_y\right)}, \tag{11}$$

но матрица формы остаётся без изменений, поскольку

$$\mathbf{M}_s = \frac{L_s}{\mathrm{P}vec(L_s)\mathrm{P}_\infty} = \frac{kL}{k\,\mathrm{P}vec(L)\mathrm{P}_\infty} = \frac{L}{\mathrm{P}vec(L)\mathrm{P}_\infty}. \tag{12}$$

Это особенно важно при реальном распознавании образов с перспективных проекций. Таким образом, можно сделать следующую классификацию преобразований:

1. Перестановка (Переиндексация вершин) $A' = P^{-1}AP = P^T AP$.
2. Параллельный перенос $\Delta = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$.
3. Вращение (Поворот) $\mathbf{P} = \mathbf{Q} = \begin{pmatrix} \cos(\varphi) & \mp\sin(\varphi) \\ \pm\sin(\varphi) & \cos(\varphi) \end{pmatrix}$.
4. Отражение $\mathbf{P} = \mathbf{Q} = I - 2uu^T$, где u – вектор единичной длины, ортогональный некоторой прямой ℓ, относительно которой происходит отражение.
5. Равномерное масштабирование $\mathbf{P} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} = kI$.
6. Растяжение / Сжатие $\mathbf{P} = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix}$.
7. Разрушающие преобразования.
8. Последовательность преобразований $\mathrm{P}=\mathrm{P}_1 \cdot \mathrm{P}_2 \neq \mathrm{P}_2 \cdot \mathrm{P}_1$ в общем виде важна, однако если матрицы коммутирующие, то порядок не важен.
9. Комбинация ортогональных и деформирующих преобразований:

- $x^T\mathbf{P}^T\mathbf{Q}^T\mathbf{Q}\mathbf{P}x = x^T\mathbf{P}^T\mathbf{P}x$ при порядке преобразований: сначала деформирующие, потом ортогональные, последние можно не учитывать;
- $x^T\mathbf{Q}^T\mathbf{P}^T\mathbf{P}\mathbf{Q}x$ Сначала ортогональные, потом деформирующие: в этом случае учитывать нужно все.

На матрицу L не влияют преобразования 1-4, матрица формы M не изменяется преобразованиями 1-5, матрица смежности A нечувствительна (в смысле сохранения смежности – изоморфизма) к преобразованиям 1-6. В общем случае в результате разрушающих преобразований меняется матрица смежности и матрица формы, но можно искать максимальное вхождение подграфа MCS, т.е. искать неразрушенные части фигуры и анализировать их деформации.

3. Переиндексация (Перестановка) графа

Хотя в геометрии редко применяют процедуры переиндексации вершин (точек), для алгоритмов на изоморфизм графа это, можно сказать, основной инструмент, поэтому подробнее распишем и зафиксируем эту процедуру. Суть переиндексации (перестановки) графа – генерация нового изоморфного графа. За отправную точку положим, что нам известна новая последовательность индексов вершин графа, а именно следующая таблица 2 (биекция), или перестановка [5].

Таблица 2. **Переиндексация вершин графа**

i	j, P^T	j, P
0 (a)	1 (**b**)	**2**
1 (b)	2 (**c**)	**0**

Окончание таблицы 2

i	j, P^T	j, P
2 (c)	0 (**a**)	**1**
3 (d)	3 (**d**)	**3**

$$\pi : \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \end{pmatrix}$$

или в виде циклической перестановки

$$\pi : (0,2,1)(3).$$

Само преобразование подобия над матрицей смежности A выражается в следующем виде:

$$A' = P^{-1}AP = P^T AP, \quad (13)$$

где P – матрица перестановок порождается перестановкой π. Учитывая, что множитель P переставляет столбцы, а P^T – строки матрицы смежности, тогда в ячейках $P[i, j] = 1$, в остальных – нули:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Обычно в решателях обходятся без этой дорогостоящей процедуры, достаточно лишь таблицы 2, полная перестановка нужна в случае, если необходимо действительно переставить вершины в графе, перед пересохранением, например.

Полный пакет преобразований над основными матрицами A,G,M будет

$$A' = P^{-1}AP = P^T AP;$$
$$G' = GP; \quad (14)$$
$$\mathbf{M}' = P^T \mathbf{M} P.$$

Эти три матрицы в теории графов играют каждая свою роль:

- A – матрица смежности, определяет топологию (структуру фигуры), размерность $n \times n$;
- G – матрица атрибутов при вершинах, характеризует свойства вершин графа, размерность $3n/2n/n = k \times n$;
- **M** – матрица реберных атрибутов, задаёт параметры связей между объектами, размерность $n \times n$.

*4. Построения фигуры по матрицам A, **M***

Опишем процесс построения фигуры воздушного змея (Kite), заданной матрицами структурной связи A и матрицей формы M. Фигура или граф, получаемый при этом (рис. 2), является некой разновидностью графа единичных кругов [14, 15] с той разницей, что круги не единичного диаметра а задаются матрицей формы, а в 3D-случае вместо кругов будут сферы.

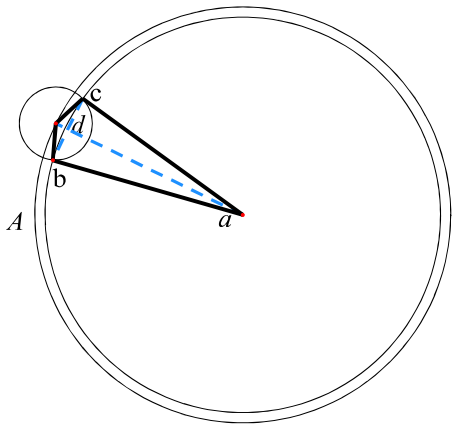

Рисунок 2. Метод окружностей

1. Зафиксируем параметры $\beta = 0.3$ и $\gamma = 0.1$, построим матрицу M и выберем нужную длину самого большого ребра $|ad| = 5$.

2. Выберем точку, от которой отходит самая длинная сторона (и/или точку с самым большим количеством ребер) *a*, и разместим её на плоскости (в любом месте, если не оговорено иное).

3. Для каждого ребра (и неявного тоже) точки *a* раствором циркуля, равным соответствующей длине, чертим окружности.

4. На окружности наибольшего радиуса (самая длинная сторона) опять же произвольно выбираем точку, у нас это будет точка *d*, внутри этой окружности и будет лежать наша фигура.

5. Из точки *d* раствором циркуля, равным длинам исходящих из этой точек ребер, чертим окружности. В нашем случае радиусы этих окружностей будут равны.

6. Точки пересечений выше построенных окружностей с окружностями (меньшего чем |ad| радиуса) с центром в точке *a* и будут искомыми точками *b* и *c*.

7. При размещении точек *b* и с нужно, если требуется, соблюдать хиральность, т.е. сохранять знак векторного произведения векторов.

Несколько замечаний

Из процесса построения видно, что система координат (как и размерные величины) никоим образом не присутствуют, напротив, сама фигура обладает характеристиками, играющими роль внутренней системы координат. Так например, определяющей величиной является размер нормирующего вектора, от которого откладываются все остальные. Основные точки можно найти, исследуя структуру матрицы *A* – граф связи, такими, скажем, могут являться максимальная клика графа, узел с максимальной степенью и т.д. В нашем случае это *a* и *d* – точки, являющиеся концами вектора наибольшей длины. Но точка *a* предпочтительней, поскольку от неё отходят наибольшие отрезки, длинные стороны ромбоида и нормирующая диагональ.

Поиск симметрий фигуры через автоморфизм графа (AUT)

```
0--------------------------------------------------
1 #Create Autocad figure from pseudo command-line example
2    # nodes setting area on graph notations
3    a = [ 0.0,-2.0]; b = [-1.0, 1.0]; c = [ 1.0, 1.0]; d = [ 0.0, 2.0];
4
5    # edges setting area on graph notations
6    line(a,b); line(b,d); line(d,c); line(c,a);
7--------------------------------------------------
8 # Mapping to graph (kite_graph.grf format example)
9 # Pattern graph
10 # Graph with vertex attributes
11    # nodes creating area  -> define $M matrix
12    4          # nodes count
13    0 0.0 -2.0 # a mapping (id + v.attr)
14    1 -1.0 1.0 # b
15    2 1.0 1.0  # c
16    3 0.0 2.0  # d
17
18 # edges creating area  -> define A matrix
19   # necessary minimum for edge storage:
20   # 0 1 2    # node 0 have connection with 1 and 2 nodes
21   # 3 1 2    # node 3 have connection with 1 and 2 nodes
22   2          # edges count from node 0
23   0 1        # line(a,b)  (id1 id2)
24   0 2        # line(a,c)
25   1          # edges count from node 1
26   1 3        # line(b,d)
27   1          # edges count from node 2
```

```
28   2 3       # line(c,d)
29   0         # edges count from node 3
30---------------------------------------------
```

Листинг 1. Отображение создания CAD-модели на граф

Для простоты обозначим $l = \sqrt{\frac{1}{(1+\gamma)^2} + \frac{\beta^2}{4}}$, $s = \sqrt{\frac{\gamma^2}{(1+\gamma)^2} + \frac{\beta^2}{4}}$ – соответственно длинные и короткие стороны ромбоида. Тогда исходные матрицы будут

$$A = \begin{matrix} \\ a \\ b \\ c \\ d \end{matrix} \begin{pmatrix} a & b & c & d \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 0 & l & l & 1 \\ l & 0 & \beta & s \\ l & \beta & 0 & s \\ 1 & s & s & 0 \end{pmatrix}.$$

Условия разрешимости можно записать в табличном виде (табл. 3), в виде матрицы или на естественном языке (табл. 4) [16].

Таблица 3. **Условия разрешимости задачи**

level	node	V.cond	E.cond
0	a	$d[0\,\vert\,2] \wedge \mathbf{M}[0\,\vert\,l,l]$	–
1	b	$d[1\,\vert\,2] \wedge \mathbf{M}[1\,\vert\,l,s]$	$A[0] \wedge \mathbf{M}[0\,\vert\,l]$
2	c	$d[2\,\vert\,2] \wedge \mathbf{M}[2\,\vert\,l,s]$	$\left(A[0] \wedge \mathbf{M}[0\,\vert\,l]\right) \wedge \left(\overline{A[1]} \wedge \mathbf{M}[1\,\vert\,\beta]\right)$
3	d	$d[3\,\vert\,2] \wedge \mathbf{M}[3\,\vert\,s,s]$	$\left(\overline{A[0]} \wedge \mathbf{M}[0\,\vert\,1]\right) \wedge \left(A[1] \wedge \mathbf{M}[1\,\vert\,s]\right) \wedge \left(A[2] \wedge \mathbf{M}[2\,\vert\,s]\right)$

При решении этой задачи посредством какого-либо алгоритма (см. обзор и библиографию [18]) на изоморфизм (с поддержкой атрибутов на ребра) получим таблицу биекций (табл. 5). Откуда формулу симметрии (без учета хиральности) можно записать в виде, схожим по своей структуре с циклической перестановкой:

$$S_e = \{a\},\{b,c\},\{d\} \longrightarrow \pi : (a)(b,c)(d).$$

Таблица 4. **Предикаты разрешимости на естественном языке**

level	node	Natural Language Predicates
0	a	для заполнения 0-го слоя нужны вершины с 2 соседями и расстоянием *l* до них
1	b	вершины с двумя соседями и с расстояниями *l*, *s* и вершина 0-го слоя имеет связь размера *l*.
2	c	вершины с двумя соседями и с расстояниями *l*, *s* и вершина 0-го слоя имеет связь размера *l* и вершина 1-го слоя имеет виртуальную связь (не имеет видимой связи) размера β.
3	d	вершины с двумя соседями и с расстоянием *s*, *s* и вершина 0-го слоя имеет виртуальную связь (не имеет видимой связи) размера 1 и вершины 1-го и 2-го слоёв имеют связи размеров *s*.

Отметим, что с точки зрения топологической симметрии (изоморфизма графа без атрибутов), симметричными будут все вершины {*a*,*b*,*c*,*d*}, а вот при учете хиральности все вершины будут уникальные {*a*}, {*b*}, {*c*}, {*d*}.

Таблица 5. **Биекции задачи на автоморфизм графа**

level	pattern node	#1	#2
0	a	a	a
1	b	b	c

Окончание таблицы 5

level	pattern node	#1	#2
2	c	c	b
3	d	d	d

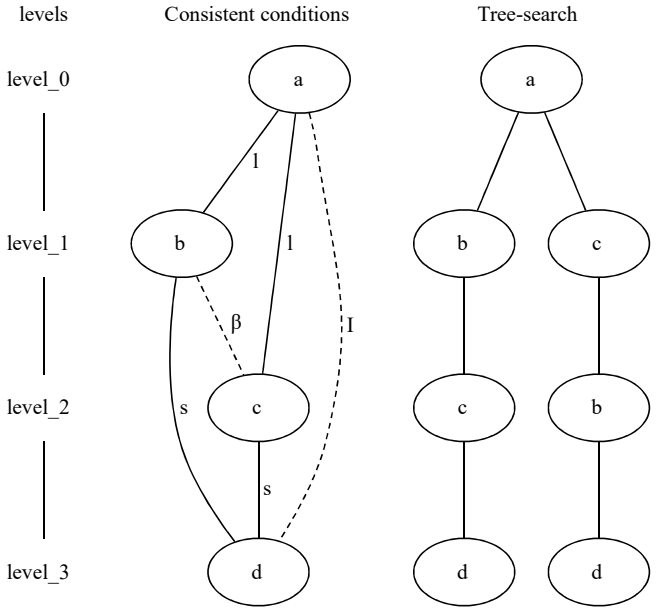

Рисунок 3. Условия разрешимости для задачи симметрии и результирующее дерево поиска

Извлечение равномерно деформированной фигуры на сложном чертеже через изоморфизм подграфа (SI)

Приведем пример извлечения заданной фигуры из неструктурированного чертежа (рис.4). В качестве искомого образца возьмем фигуру kite (см. рис. 1) с матрицей координат

$$G_{kite} = \begin{pmatrix} a & b & c & d \\ 0 & -1 & 1 & 0 \\ -2 & 1 & 1 & 2 \end{pmatrix}.$$

На рисунке этот образец разместим с помощью следующих преобразований:

$$G_{kite}^{SI} = \begin{pmatrix} 11 \\ 15 \end{pmatrix}. + Q \cdot P \cdot \left(\begin{pmatrix} 0 \\ 2 \end{pmatrix}. + G_{kite} \right) = \begin{pmatrix} 11.0 & 11.5072 & 12.8928 & 12.6 \\ 15.0 & 17.4785 & 16.6785 & 17.7713 \end{pmatrix},$$

где матрица вращения $Q = \begin{pmatrix} cos(\varphi) & sin(\varphi) \\ -sin(\varphi) & cos(\varphi) \end{pmatrix} = \begin{pmatrix} 0.866025 & 0.5 \\ -0.5 & 0.866025 \end{pmatrix}$ при $\varphi = -30^{\circ}$ и матрица равномерного скалирования $P = \begin{pmatrix} 0.8 & 0 \\ 0 & 0.8 \end{pmatrix}$. И с помощью алгоритма на изоморфизм подграфа мы должны найти эту комбинацию точек их взаимного расположения среди всех точек и линий чертежа. Размещение фигуры можно производить не по координатам вершин, а например, с помощью графового метода дисков заданного радиуса [14, 15], где в качестве точки отсчета можно взять точку *pt*17 = [11.0;15.0].

Сам чертеж имеет простую структуру (рис. 5), состоящую из 9 компонент связности, хотя, по правилам, принятым в конструкторских фирмах, структура может несколько меняться, всё зависит от того, как рисуют пересечение линий и точки на линиях. Если точка делит линию, то задача становится сложной из-за возникающего гомеоморфизма.

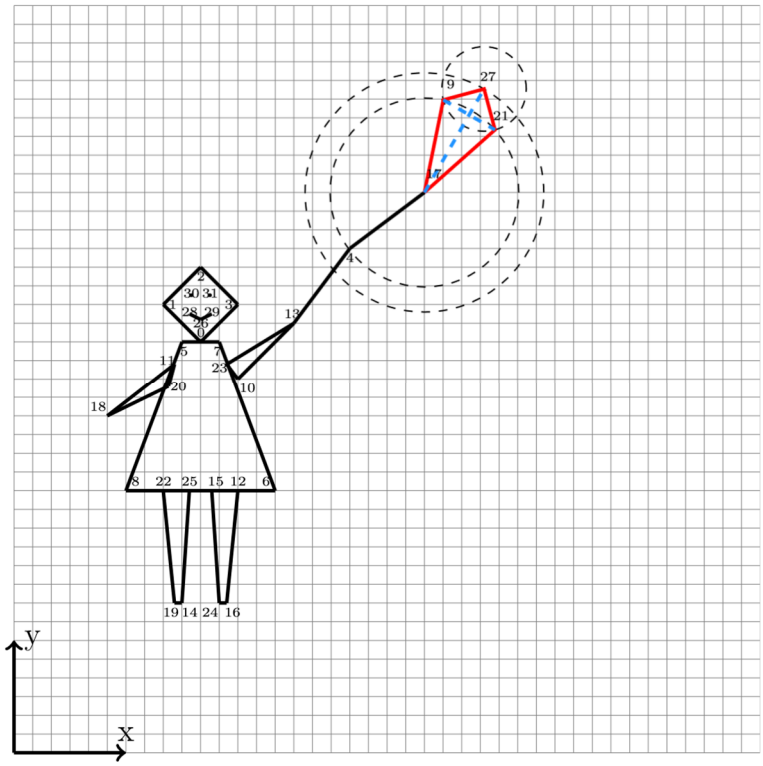

Рисунок 4. Эпическое полотно ’Kate and Kite’

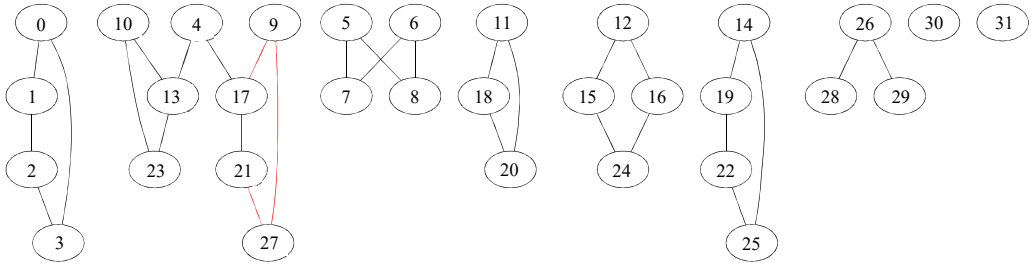

Рисунок 5. Графовое (структурное) представление чертежа без атрибутов

```
0 Solution count: 40
1  pattern size: 4
2 ------------------------------------
3 Total states exploration: 134
4 Total states handled: 174
5 --- Breeding factor statistics ---
6 0:  |28: 1 | width: 28, average value: 28 disp: 0
7 1:  |0: 1  |2: 25 |3: 2 | width: 56, average value: 2 disp: 0.21428573
8 2:  |0: 10 |1: 42 |2: 4 | width: 50, average value: 0.89285713 disp: 0.23852043
9 3:  |0: 10 |1: 40 | width: 40, average value: 0.80000001 disp: 0.16
10 Total Width = 174
11 0:27|27|25|25|24|24|22|22|21|21|19|19|17|17|16|16|15|15|14|14|12|12|9|9|8|8|7|7|6|6|5|5|3|3|2|2|1|1|0|0
12 1:21|9|22|14|16|15|25|19|27|17|22|14|21|9|24|12|24|12|25|19|16|15|27|17|6|5|6|5|8|7|8|7|2|0|3|1|2|0|3|1
13 2:9|21|14|22|15|16|19|25|17|27|14|22|9|21|12|24|12|24|19|25|15|16|17|27|5|6|5|6|7|8|7|8|0|2|1|3|0|2|1|3
14 3:17|17|19|19|12|12|14|14|9|9|25|25|27|27|15|15|16|16|22|22|24|24|21|21|7|7|8|8|5|5|6|6|1|1|0|0|3|3|2|2
15 Solutions after symmetry: 5
16   1: {27,17,21,9}
17   2: {25,22,14,19}
18   3: {24,16,15,12}
19   4: {8,6,5,7}
20   5: {3,2,0,1}
```

Листинг 2. Вывод алгоритма изоморфизма подграфа

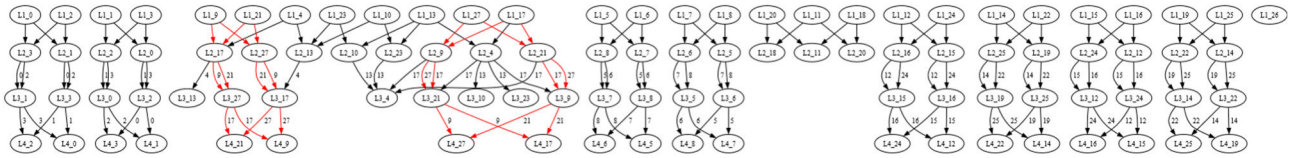

Рисунок 6. Типовой дамп дерева поиска решателями 1-го поколения (алг.1)

Таблица 6. **Список найденных четырехугольников и их матрицы формы**

#	Quadrilaterals	**M** matrix	Shape
0	{**27,17,21,9**}	$\begin{pmatrix} 0.0 & 1.0 & 0.3536 & 0.3536 \\ 1.0 & 0.0 & 0.7906 & 0.7906 \\ 0.3536 & 0.7906 & 0.0 & 0.5 \\ 0.3536 & 0.7906 & 0.5 & 0.0 \end{pmatrix}$	Kite
1	{25,22,14,19}	$\begin{pmatrix} 0.0 & 0.3972 & 0.9827 & 1.0 \\ 0.3972 & 0.0 & 0.9665 & 0.9581 \\ 0.9827 & 0.9665 & 0.0 & 0.0636 \\ 1.0 & 0.9581 & 0.0636 & 0.0 \end{pmatrix}$	Trapezoid
2	{24,16,15,12}	$\begin{pmatrix} 0.0 & 0.0636 & 0.9827 & 0.9665 \\ 0.0636 & 0.0 & 1.0 & 0.9581 \\ 0.9827 & 1.0 & 0.0 & 0.3972 \\ 0.9665 & 0.9581 & 0.3972 & 0.0 \end{pmatrix}$	Trapezoid
3	{8,6,5,7}	$\begin{pmatrix} 0.0 & 0.848 & 0.9057 & 1.0 \\ 0.848 & 0.0 & 1.0 & 0.9057 \\ 0.9057 & 1.0 & 0.0 & 0.212 \\ 1.0 & 0.9057 & 0.212 & 0.0 \end{pmatrix}$	Trapezoid
4	{3,2,0,1}	$\begin{pmatrix} 0.0 & 0.7071 & 0.7071 & 1.0 \\ 0.7071 & 0.0 & 1.0 & 0.7071 \\ 0.7071 & 1.0 & 0.0 & 0.7071 \\ 1.0 & 0.7071 & 0.7071 & 0.0 \end{pmatrix}$	Square

1. Пример анализа преобразований фигуры

В качестве примера опишем процедуру восстановление к. масштабирования (в нашем случае он взят 0.8 по построению), поскольку он наиболее важен, в перспективных проекциях, через него определяется расстояние до объекта. Так, после задачи GI с атрибутами на ребра алгоритма 1 и учета симметрии графа шаблона $S_e = \{a\}\{b,c\}\{d\}$ получим искомую таблицу 7 биекций с выровненным порядком индексации вершин. Матрица формы фигуры на чертеже, очевидно, будет равна аналогичной матрице паттерн графа, а вот матрица длин в случае скалирования будет другая.

Таблица 7. **Искомые биекции алгоритма 1**

levels	ptr	trg
0	a	17
1,2	b,c	9,21
3	d	27

$$L_{kite}^{SI} = \rho\_matrix([pt17, pt9, pt21, pt27]) = \begin{pmatrix} 0.0 & 2.5298 & 2.5298 & 3.2 \\ 2.5298 & 0.0 & 1.6 & 1.1314 \\ 2.5298 & 1.6 & 0.0 & 1.1314 \\ 3.2 & 1.1314 & 1.1314 & 0.0 \end{pmatrix}.$$

Видно, что матрица длин отличается от исходной (3) на множитель 0.8, для его нахождения достаточно взять отношение максимальной дины в найденной форме к максимальной длине оригинальной фигуры $\frac{3.2}{4} = 0.8$ или любых других соответствующих элементов, кроме диагональных.

```
0 ------------------------------------------------
1 #Create AutoCAD figure from pseudo command-line example
2     # nodes setting area on graph notations
3     pt0=[5.0,11.0];      pt1=[4.0,12.0];          pt2=[5.0,13.0];      pt3=[6.0,12.0];
4     pt4=[9.0,13.5];      pt5=[4.5,11.0];          pt6=[7.0,7.0];       pt7=[5.5,11.0];
5     pt8=[3.0,7.0];       pt9=[11.50718,17.47846]; pt10=[6.0,10.0];     pt11=[4.3,10.4];
6     pt12=[6.0,7.0];      pt13=[7.5,11.5];         pt14=[4.5,4.0];      pt15=[4.75,7.0];
7     pt16=[5.7,4.0];      pt17=[11.0,15.0];        pt18=[2.5,9.0];      pt19=[4.3,4.0];
8     pt20=[4.14,9.8];     pt21=[12.89282,16.67846]; pt22=[4.0,7.0];     pt23=[5.7,10.4];
9     pt24=[5.5,4.0];      pt25=[5.25,7.0];         pt26=[5.0,11.6];     pt27=[12.6,17.77128];
10    pt28=[4.7,11.75];    pt29=[5.3,11.75];        pt30=[4.75,12.25];   pt31=[5.25,12.25];
11
12    # edges setting area on graph notations
13      # head
14
15    line(pt0,pt1);line(pt1,pt2);line(pt2,pt3);
16    line(pt3,pt0);line(pt28,pt26);line(pt26,pt29);
17      # body
18    line(pt5,pt7);line(pt7,pt6);line(pt6,pt8);line(pt8,pt5);
19      # right hand
20    line(pt18,pt11);line(pt11,pt20);line(pt20,pt18);
21      # left hand
22    line(pt13,pt10);line(pt10,pt23);line(pt23,pt13);
23      # right leg
24    line(pt19,pt14);line(pt14,pt25);line(pt25,pt22);line(pt22,pt19);
25      # left leg
26    line(pt24,pt16);line(pt16,pt12);line(pt12,pt15);line(pt15,pt24);
27      # rope
28    line(pt13,pt4);line(pt4,pt17);
29      # kite
30    line(pt17,pt21);line(pt21,pt27);line(pt27,pt9);line(pt9,pt17);
31 ------------------------------------------------
```

Листинг 3. CAD-модель и её граф

Data: $H_p(A_p, G_p), H_t(A_t, G_t)$
Result: $B^{A,M}_{(V_p \rightarrowtail V_t)}, B^{A}_{(V_p \rightarrowtail V_t)}$; /* Struct+Semantic and only Struct Bijection tables */

$M_p \leftarrow M\_matrix(G_p)$;
$H_p(A_p, G_p) \leftarrow M_p$;
$C \leftarrow ConsistentConditions(H_p(A_p, G_p, M_p))$;
$Matchings \leftarrow SI(C, H_t(A_t))$; /* Subgraph Isomorphism (not e.attributed) task */
while $size(Matchings) \neq 0$ do
  $H_s(A_s, G_s) \leftarrow Matchings.pop()$;
  $M_s \leftarrow M\_matrix(G_s)$;
  $H_s(A_s, G_s) \leftarrow M_s$;
  $S \leftarrow GI(C, H_s(A_s, M_s))$; /* Graph Isomorphism (e.attributed) task */
  if $S \neq 0$ then
    $B^{A,M}_{(V_p \rightarrowtail V_t)}.push(S)$; /* Struct & Semantic solutions */
  else
    $B^{A}_{(V_p \rightarrowtail V_t)}.push(H_s(A_s))$; /* Struct only solutions */
  end
end

Алгоритм 1. Алгоритм разделённого решения

2. Графы отношений высших порядков

Матрица M не единственная, что может описать форму, в некоторых задачах может применяться матрица *L*: например, когда размеры объекта известны и не меняются. Но всё-таки больший интерес вызывает задача классификации форм, т.е. выявление отличий, например, ромба от kite или от квадрата и распознавания именно формы фигуры (например kit'ы разных пропорций). Тематика этой статьи ограничена постановкой задачи извлечения форм, а то, что сказано ниже, уже относится к сфере искусственного интеллекта, т.е. алгоритмам логической классификации объектов и выявления общих черт. Приведем здесь только методику построения графов отношений, которые уже применимы для автоматического построения логических предикатов и последующего их решения.

Для фигуры из четырех углов (n = 4) мы имеем $e = \frac{n(n-1)}{2} = 6$ ребер, 4 видимых и 2 неявных (диагонали). Сформируем из них также матрицу отношений (записанную в лексикографическом порядке):

$$\Lambda = \begin{pmatrix} 1 & \mathrm{P}\vec{ac}\,\mathrm{P}_2\mathrm{P}\mathbf{ab}\,\mathrm{P}_2 & \mathrm{P}\vec{ad}\,\mathrm{P}_2\mathrm{P}\mathbf{ab}\,\mathrm{P}_2 & \mathrm{P}\vec{bc}\,\mathrm{P}_2\mathrm{P}\mathbf{ab}\,\mathrm{P}_2 & \mathrm{P}\vec{bd}\,\mathrm{P}_2\mathrm{P}\mathbf{ab}\,\mathrm{P}_2 & \mathrm{P}\vec{cd}\,\mathrm{P}_2\mathrm{P}\mathbf{ab}\,\mathrm{P}_2 \\ \mathrm{P}\vec{ab}\,\mathrm{P}_2\mathrm{P}\vec{ac}\,\mathrm{P}_2 & 1 & \mathrm{P}\vec{ad}\,\mathrm{P}_2\mathrm{P}\vec{ac}\,\mathrm{P}_2 & \mathrm{P}\vec{bc}\,\mathrm{P}_2\mathrm{P}\vec{ac}\,\mathrm{P}_2 & \mathrm{P}\vec{bd}\,\mathrm{P}_2\mathrm{P}\vec{ac}\,\mathrm{P}_2 & \mathrm{P}\vec{cd}\,\mathrm{P}_2\mathrm{P}\vec{ac}\,\mathrm{P}_2 \\ \mathrm{P}\vec{ab}\,\mathrm{P}_2\mathrm{P}\mathbf{ad}\,\mathrm{P}_2 & \mathrm{P}\vec{ac}\,\mathrm{P}_2\mathrm{P}\mathbf{ad}\,\mathrm{P}_2 & 1 & \mathrm{P}\vec{bc}\,\mathrm{P}_2\mathrm{P}\mathbf{ad}\,\mathrm{P}_2 & \mathrm{P}\vec{bd}\,\mathrm{P}_2\mathrm{P}\mathbf{ad}\,\mathrm{P}_2 & \mathrm{P}\vec{cd}\,\mathrm{P}_2\mathrm{P}\mathbf{ad}\,\mathrm{P}_2 \\ \mathrm{P}\vec{ab}\,\mathrm{P}_2\mathrm{P}\vec{bc}\,\mathrm{P}_2 & \mathrm{P}\vec{ac}\,\mathrm{P}_2\mathrm{P}\vec{bc}\,\mathrm{P}_2 & \mathrm{P}\vec{ad}\,\mathrm{P}_2\mathrm{P}\vec{bc}\,\mathrm{P}_2 & 1 & \mathrm{P}\vec{bd}\,\mathrm{P}_2\mathrm{P}\vec{bc}\,\mathrm{P}_2 & \mathrm{P}\vec{cd}\,\mathrm{P}_2\mathrm{P}\vec{bc}\,\mathrm{P}_2 \\ \mathrm{P}\vec{ab}\,\mathrm{P}_2\mathrm{P}\vec{bd}\,\mathrm{P}_2 & \mathrm{P}\vec{ac}\,\mathrm{P}_2\mathrm{P}\vec{bd}\,\mathrm{P}_2 & \mathrm{P}\vec{ad}\,\mathrm{P}_2\mathrm{P}\vec{bd}\,\mathrm{P}_2 & \mathrm{P}\vec{bc}\,\mathrm{P}_2\mathrm{P}\vec{bd}\,\mathrm{P}_2 & 1 & \mathrm{P}\vec{cd}\,\mathrm{P}_2\mathrm{P}\vec{bd}\,\mathrm{P}_2 \\ \mathrm{P}\vec{ab}\,\mathrm{P}_2\mathrm{P}\vec{cd}\,\mathrm{P}_2 & \mathrm{P}\vec{ac}\,\mathrm{P}_2\mathrm{P}\vec{cd}\,\mathrm{P}_2 & \mathrm{P}\vec{ad}\,\mathrm{P}_2\mathrm{P}\vec{cd}\,\mathrm{P}_2 & \mathrm{P}\vec{bc}\,\mathrm{P}_2\mathrm{P}\vec{cd}\,\mathrm{P}_2 & \mathrm{P}\vec{bd}\,\mathrm{P}_2\mathrm{P}\vec{cd}\,\mathrm{P}_2 & 1 \end{pmatrix} \quad (15)$$

и конкретное численное воплощение kite-матрицы:

$$\Lambda_{\Diamond} = \begin{pmatrix} 1.0 & \mathbf{1.0} & 1.2649 & 0.6325 & 0.4472 & 0.4472 \\ \mathbf{1.0} & 1.0 & 1.2649 & 0.6325 & 0.4472 & 0.4472 \\ 0.7906 & 0.7906 & 1.0 & \mathbf{0.5} & 0.3536 & 0.3536 \\ 1.5811 & 1.5811 & \mathbf{2.0} & 1.0 & \mathbf{0.7071} & \mathbf{0.7071} \\ 2.2361 & 2.2361 & 2.8284 & \mathbf{1.4142} & \mathbf{1.0} & \mathbf{1.0} \\ 2.2361 & 2.2361 & 2.8284 & \mathbf{1.4142} & \mathbf{1.0} & 1.0 \end{pmatrix}. \quad (16)$$

По числам видно, какие стороны равны (1 в позиции), какие имеют целые отношения ($\frac{1}{\beta} = 2$), а также скрытые отношения, которые сразу не видно – ($1.41421 \approx \sqrt{2} = \frac{\mathrm{P}\vec{bc}\,\mathrm{P}_2}{\mathrm{P}\vec{cd}\,\mathrm{P}_2}$) – гипотенуза равнобедренного прямоугольного треугольника $\triangle bdc$. Для квадрата, например, $\Lambda\square$ будет куда более выразительна:

$$\Lambda_{\mathrm{W}} = \begin{pmatrix} 1.0 & 1.0 & 1.4142 & 1.4142 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.4142 & 1.4142 & 1.0 & 1.0 \\ 0.7071 & 0.7071 & 1.0 & 1.0 & 0.7071 & 0.7071 \\ 0.7071 & 0.7071 & 1.0 & 1.0 & 0.7071 & 0.7071 \\ 1.0 & 1.0 & 1.4142 & 1.4142 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.4142 & 1.4142 & 1.0 & 1.0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & \sqrt{2} & \sqrt{2} & 1 & 1 \\ 1 & 1 & \sqrt{2} & \sqrt{2} & 1 & 1 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 & 1 & 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 & 1 & 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & 1 & \sqrt{2} & \sqrt{2} & 1 & 1 \\ 1 & 1 & \sqrt{2} & \sqrt{2} & 1 & 1 \end{pmatrix} \quad (17)$$

Видно, что матрица Λ уже не симметрическая, верхний надтреугольник состоит из прямых отношений, нижний – из обратных (такую матрицу можно назвать, например, обратно-симметрической,

$\Lambda \circ \Lambda^T = \Lambda^T \circ \Lambda = J$, J -матрица единиц/единичный элемент в произведении Адамара). Также видно, что наша матрица M (делитель в элементах выделен синим) является лишь подмножеством более общей матрицы Λ. Основным свойством матрицы является то, что она состоит из соотношений всех размеров в фигуре (отсылая к пифагорейской математике, уходящей вглубь вавилонской алгебры [9]) и единичные элементы будут отвечать равным отрезкам, а если мы будем использовать удвоенную мировую функцию, т.е. $2\sigma = \rho^2$, то 2 (двойка) в определенной позиции означает гипотенузу равнобедренного прямоугольного треугольника (диагональ квадрата). Матрицы $\mathbf{M}, \Lambda$ пригодны для использования в системах компьютерной алгебры – CAS. С последней ассоциирована своя матрица смежности A_Λ, но её появление для текущих задач несколько преждевременно.

2.1. Построение матрицы Λ

$$L_h = L_n vec(L^T) = vech(L) = \left(\mathrm{P}\vec{a}b\,\mathrm{P}_2 \quad \mathrm{P}\vec{a}c\,\mathrm{P}_2 \quad \mathrm{P}\vec{a}d\,\mathrm{P}_2 \quad \mathrm{P}\vec{b}c\,\mathrm{P}_2 \quad \mathrm{P}\vec{b}d\,\mathrm{P}_2 \quad \mathrm{P}\vec{c}d\,\mathrm{P}_2\right)^T \tag{18}$$

где $L_n = \begin{pmatrix} 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \end{pmatrix}$ – матрица исключения (elimination matrix) [17]. Тогда матрица Λ строится уже через матрицу L_h и вектор единиц j соответствующего размера

$$\Lambda = diag(L_h)^{-1} \cdot (j \otimes L_h^T). \tag{19}$$

3. Совместное решение и динамическая нормировка

При решении задачи выделения формы посредством SI алгоритма с одновременным поиском нужной формы, нам наперед не известен нормирующий множитель, чтобы из размерной матрицы L получить безразмерную $\mathbf{M}$, поэтому можно проводить нормировку динамически, используя для этого первую длину ребра от слоя 0 к 1 (множитель α), особенно это актуально при решении задачи MCS (пересечения графов). Основу для этого как раз обеспечивает матрица Λ паттерн графа.

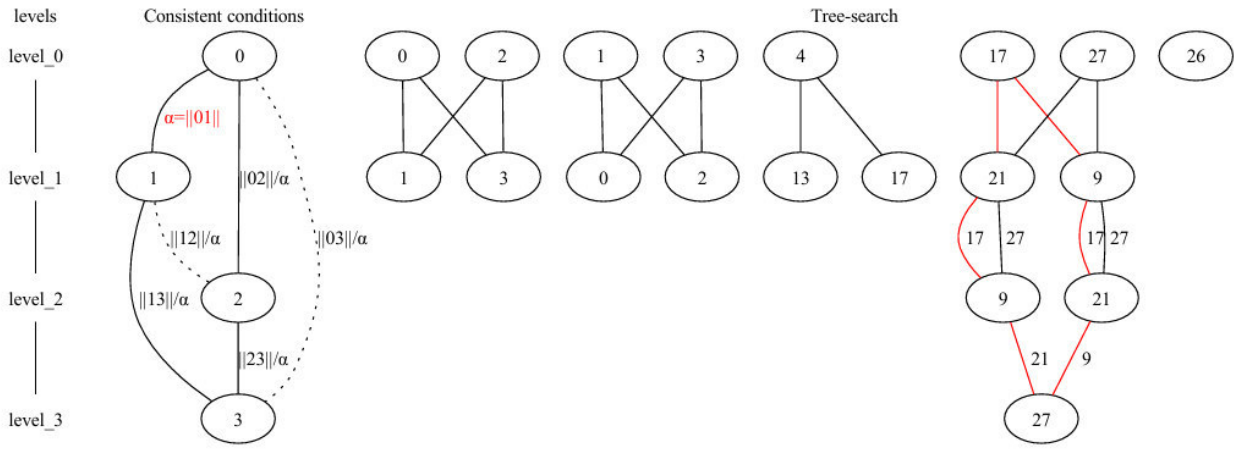

Рисунок 7. Условия разрешимости для обобщенно-взвешенного SI алгоритма и его дерево поиска

Data: $H_p(A_p, G_p), H_t(A_t, G_t)$
Result: $B^{A,M}_{(V_p \mapsto V_t)}$; /* Bijection table or reduction tree */
1 $C \leftarrow ConsistentConditions(H_p(A_p, G_p))$;
2 $\alpha \leftarrow C.getLeaderEdgeAttribute()$;
3 $\rho_p \leftarrow \rho\_matrix(G_p)$;
4 $M_p \leftarrow M\_matrix(\rho_p, \alpha)$;
5 $H_p(A_p, G_p) \leftarrow M_p$;
6 $C \leftarrow C.update(H_p(A_p, G_p, M_p))$;
7 $\rho_t \leftarrow \rho\_matrix(G_t)$;
8 $H_t(A_t, G_t) \leftarrow \rho_t$;
9 $B^{A,M}_{(V_p \mapsto V_t)} \leftarrow SI(C, H_t(A_t, \rho_t))$; /* Subgraph Isomorphism (e.attributed) task */

Алгоритм 2. Алгоритм совместного решения

Таблица 8. **Сравнения алгоритмов решения**

	Алгоритм 1	Алгоритм 2
Память	Не нужна большая матрица *L*/(M)	Строит большую матрицу *L*
Дерево поиска	Велико	Малое
Быстродействие	Умеренное	Велико
Обход дерева	Bfs/Dfs	Dfs like
Модель параллелизма	SI:parallel for/GI:Tasks по формам	Tasks по деревьям леса
Вложенный параллелизм	-/+	+
Недостатки	Множественные решения	Сложности с SIMD-конвейером
Достоинства	Декомпозирует модель на формы	Быстро находит решение
Использование	Анализ и исследование	Промышленное

A. Сравнение графовых подходов и методов машинного обучения

Таблица 9. **Сравнение методов**

	Машинное обучение	Графовые предикаты
Типы мышления	Рефлексивное мышление	Абстрактно-логическое
Скорость работы	Высокая	В общем случае низкая
Точность результатов	Вероятностная оценка	Точный метод
Детерминизм	Нет	Строгий
Предпочтительный способ реализации	Аналоговый (мозг)	Цифровой (ЭВМ)
Предсказательная способность	Низкая	Высокая
Обучение	Необходимо	Нет
Способ развития	Дообучение	Добавления/вывод предиката
Передача знаний	Обучение с учителем	Копирование свода предикатов
Абстракция	Слабая	Абстрактно по сути
Энергозатраты	Низкие	Высокие
Параллелизм	Высокий	В общем случае низкий
Место в когнитивном каскаде	Первичная обработка	Уточнение выхода нейронной сети

B. Julia код для построения основных матриц статьи

```
# form.jl Functions for form matrix calculation
# include("c:\\LOCAL\\julia\\form.jl")
using LinearAlgebra

function σ_matrix(G)
  dim = size(G)
  σ = zeros(dim[2],dim[2])
```

```
  Vi = zeros(dim[2],dim[2])
  for i in 1:dim[1]
    for j in 1:dim[2]  # Vi = kron(j,G[i,:]) analog
      Vi[j,:] = G[i,:]
    end
    Vi -= Vi'
    Vi .*= Vi
    σ += Vi
  end
  return σ
end

function ρ_matrix(G)
  ρ = sqrt.(σ_matrix(G))
  return ρ
end
function M_matrix(G)
  ρ = ρ_matrix(G)
  # k = norm(vec(ρ), Inf)
  k = maximum(vec(ρ))
  M = (1/k).*ρ
  return M
end

function M_matrix(ρ,α)
  M = (1/α).*ρ
  return M
end

function Λ_matrix(G)
  ρ = ρ_matrix(G)
  dim = size(G)
  n = dim[2]
  k = convert(Integer,(n*(n-1))/2)
  Λ = zeros(k,k)
  Lh = zeros(k,1)
  iter = 1
  for i in 1:n
    for j in i+1:n
      Lh[iter,] = ρ[i,j]
      iter += 1
    end
  end
  D = (inv(Diagonal(vec(Lh))))
  Λ = D*kron(ones(k,1),Lh')
  return Λ
end
function M_matrix(Λ,line)
  n = convert(Integer,(1+sqrt(1+8*size(Λ)[1]))/2)
  M = zeros(n,n)
  iter = 1
  for i in 1:n
    for j in i+1:n
      if i != j
        M[i,j] = Λ[line,iter]
        M[j,i] = M[i,j]
        iter += 1
      end
```

```
    end
    return M
end

function permutation(new_order)
  n = size(new_order)[1]
  P = zeros(n,n)
  for i in 1:n
    P[i,new_order[i]] = 1;
  end

  return P
end

# calling example:
julia> include("c:\\LOCAL\\julia\\form.jl")
M_matrix (generic function with 1 method)
julia> G=[0 -1 1 0; -2 1 1 2];
julia> M = M_matrix(G);
julia> display(M)
 0.0       0.790569  0.790569  1.0
 0.790569  0.0       0.5       0.353553
 0.790569  0.5       0.0       0.353553
 1.0       0.353553  0.353553  0.0
```

Листинг 4: Листинги на языке Julia

C. Онлайн-ресурсы

- Bit Twiddling Hacks
- Permutations as a Product of Transpositions
- TUDataset: A collection of benchmark datasets for learning with graphs
- The Network Data Repository with Interactive Graph Analytics and Visualization

Благодарности

Автор благодарит компанию *Navio* за финансирование участия в конференции и лично *Наталию Анатольевну Францеву* за вклад в подготовку доклада.

Список литературы

1. Menger K. Untersuchen uber allgemeine Metrik // Mathematische Annalen. Vol. 100. P. 75-113, 1928.
2. Blumenthal L.M. Theory and applications of Distance Geometry. Oxford: Clarendon Press, 1953. 368 p.
3. Liberti L., Lavor C. Six mathematical gems from the history of Distance Geometry //Special Issue:Many Faces of Distances. 2016. Vol. 23. iss. 5. arXiv:1502.02816
4. Рылов Ю.А. Дискретная геометрия пространства-времени и каркасная концепция динамики частиц. URL: ggdgp3rw.pdf
5. Rotman J.J. A First Course in Abstract Algebra with Applications. 3rd edition. Upper Saddle River: Prentice Hall, 2006. 581 p.
6. Rylov Yu. A. (2011) Discrete space-time geometry and skeleton conception of particle dynamics // International Journal of Theoretical Physics. 2012. Vol. 51, iss. 6. P. 1847-1865. DOI: 10.1007/s10773-011-1061-y. Available at: arXiv:1110.3399v1
7. Rylov Yu.A. (2001) Geometry without topology as a new conception of geometry // Int. Jour. Mat. & Mat. Sci. 2002. Vol. 30, iss. 12. P. 733-760. Available at: arXiv:math/0103002.
8. Рылов Ю.А. Что такое геометрия? Математическая геометрия и физическая геометрия как два аспекта собственно евклидовой геометрии. URL: tg1rw.pdf
9. Ван Дер Варден Б.Л. Пробуждающаяся наука I. Математика древнего Египта, Вавилона и Греции. М.: ГИФМЛ, 1959. 460 с.
10. Synge J.L. Relativity: the General Theory. Amsterdam: North-Holland Publishing Company, 1960
11. Mac Duffee C.C. The Theory of Matrices. Springer, Berlin 1933.

12. Ланкастер П. Теория матриц. М.: Наука, 1973. 280 с.

13. Хорн Р., Джонсон Ч. Матричный анализ. М.: Мир, 1989. 656 с.

14. Clark B. N. Unit disk graphs, M. Math. Thesis, University of Waterloo, 1985.

15. Clark B. N., Colbourn C. J., Johnson D. S. Unit disk graphs //Discrete mathematics. 1990. Vol. 86, no. 1-3. P. 165-177.

16. Boole G. An Investigation of the Laws of Thought. New York: Dover Publications, 1854.

17. Magnus, J. R., Neudecker, H. The Elimination Matrix: Some Lemmas and Applications // SIAM Journal on Algebraic Discrete Methods. 1980. Vol. 1, no. 4. P. 422–449. doi:10.1137/0601049

18. Федотов И.А. Особенности алгоритмической и архитектурной оптимизации алгоритмов сопоставления графов на примере задачи изоморфного вхождения подграфа// Графы, игры и модели: материалы междунар. науч. конф., 14 окт. 2022 г. Майкоп: АГУ, 2022. С. 42-46. URL: https://ggm.cmcagu.ru/

19. Федотов И.А. Вероятностная оценка ширины дерева поиска в задаче изоморфного вхождения подграфа// Вероятностные методы в дискретной математике: материалы XI междунар. науч. конф., 29 мая 2024 г. Петрозаводск: ИПМИ КарНЦ РАН, 2024. С. 118-121. URL: http://vmdm2024.krc.karelia.ru