Практическая задача разрезания графа

Введение

Рассматривая задача является вариаций задач о гомоморфизм графов, разбиение графа, раскраски. Подобные задачи возникают в управление кластерами, выполняющими процессы, обменивающимися данными, при тестировании микросхем на массово параллельных системах. В зависимости от области применения в постановке могут отличаться размеры входных данных, требования к скорости построения (приближённого) решения, конкретная форма целевой и ограничений.

Применительно к задаче тестирования микросхем графовая постановка не совсем точно отражает физические свойства этих самых микросхем, поэтому задача формулировался в терминах гиперграфа. Здесь мы опишем чисто графовую постановку, так как, на наш взгляд, она имеет практическое значение, интересна, содержит основные трудности, которые есть гиперграфой постановке, может служить приближённым или начальным решением для гиперграфовой и не содержит экзотических и не совсем однозначно описываемых понятий, как "ориентированные гипердуги". Также из постановки исключены веса, которые могут предоставит, как у вершин, так и ребёр, так как данных для таких весов предоставить не сможем.

Обозначения

Пусть $G_1 = (V_1, E_1)$ — полный ориентированный граф, описывающий кластер, $G_2 = (V_2, E_2)$ — некоторый разреженный ориентированный, описывающий работы, которые нужно выполнить на кластере. Для вершин графа G_1 заданы ёмкости:

$$w(\cdot) : V_1 \to \mathbb{N}.$$

Предполагается, что суммарная ёмкость узлов G_1 не меньше число вершин графа G_2:

$$\sum_{n \in V_1} w(n) \geq |V_2|.$$

Постановка

Необходимо отобразить вершины графа G_2 в узлы графа G_1 так, чтобы суммарное число вершин G_2, отображённых в узел G_1, не должно превышать ёмкость узлы G_1, т.е. построить отображение $l(\cdot) : V_2 \to V_1$ такое, что

$$|l^{-1}(n)| \leq w(n) \ \ \forall n \in V_1, l^{-1}(n) = \{v \in V_2 | l(v) = n\}.$$

При этом требуется минимизировать некий дополнительный критерий относительно распределения дуг графа G_2. Варианты этого критерия могут быть следующие:

1. Суммарную загрузку рёбер графа G_1 рёбрами графа G_2:
$$|\{(v', v'') \in E_2 | l(v') \neq l(v'')\}|.$$
2. Максимальную загрузку рёбер графа G1 рёбрами графа G2:
$$\max_{n_1 \neq n_2, n_1 \in V_1, n_2 \in V_1} |\{(v', v'') \in E_2 | l(v') = n_1, \, l(v'') = n_2\}|.$$
3. Максимальную исходящую и(или входящую, или максимум из них) загрузку узлов G_1:
$$\max_{n \in V_1} |\{(v', v'') \in E_2 | l(v') = n, \, l(v'') \neq n\}|.$$
4. Максимальную стоимость вершин G_2, вычисляемую как число инцидентных ребер, которые проходят по ребрам G_1:
$$\max_{v' \in V_2} |\{(v', v'') \in E_2 | l(v') \neq l(v'')\}|.$$

Сложность решения

Исходная задача NP-полная. Получить точное решение для графов с числом вершин больше 1000 представляется сомнительным мероприятием. Но с другой стороны, поиск точных алгоритмов, которые можно применить к небольшим графам имеет смысл и может рассматриваться в качестве одной из постановок. Референсный точный алгоритм можно получить через сведение задачи к ЦЛП.

В практических задачах либо временные рамки, либо размеры графов таковы, что нахождение точных решений не представляется возможным. И искать приближённое решение предлагается с помощью эвристик. Можно искать общие эвристики для достаточно широкого класса графов, можно пытать выделить какую-то группу и как-то учесть свойства графов в ней для создания более эффективных эвристик.

В теории, можно искать для разных типов сложности разные решения. С точки зрения практики, алгоритмы сложнее чем $O((|V_2| + |E_2|)^2|V_1|)$ малопригодны.

Данные

В качестве входных данный для задачи в качестве G_2 предлагается брать графы из набора A Benchmark Set for Multilevel Hypergraph Partitioning Algorithms.

В наборе описаны гиперграфы. Для получения графов предлагается рассматривать гипердугу $(v_1, v_2, \ldots, v_n)$ как набор ориентированных дуг

$$(v_1, v_2), (v_1, v_3), \cdots, (v_1, v_n).$$

В этом наборе можно отбирать разные типы графов по названиям и рассматривать поведение подходов на определённых заданных типах графов.

G_1 полный. Число его узлов можно взять от 4 до нескольких тысяч. Ёмкость узлов G_1 можно считать одинаковой

$$w(n) = (1 + \varepsilon)|V_2|/|V_1|,$$

для некоторого $\varepsilon > 0$.

Быстрый базовый алгоритм

Алгоритм последовательно проходит вершины V_2 и выбирает для неё расположение среди узлов V_1. Путь на k-ом шаге $l_k(\cdot) : V_2 \to V_1$ расположение первых $k-1$ вершины G_2. Путь $O(v)$ все вершины смежные с v.

1. Для $v_k \in V(G_2)$ для всех $v \in O(v_k)$ считаем список частот F узлов $l(v)$, вмещающих v_k:

$$F = \{(n, c) : w(n) > |l^{-1}(n)|, c = |\{l^{-1}(n) \cap O(v_k)\}|\}.$$

2. Если F не содержит не нулевых c приписываем v_k равномерно случайной узлу среди узлов в F. Переходим к к следующий вершине G_2.
3. Генерируем узел как реализацию случайного распределение с частотами F. Переходим к следующей вершине G_2.

Существующие решения

Существует набор готовых открытых и закрытых решений для подобной задачи. Их можно использовать по-разному. Можно, сравниваться с ними по скорости и качеству, можно сравнивать разные решения между собой, можно изучать, как они устроенны, можно использовать в качестве решателей надстраивая вокруг что-то своё. В частности

- KaHIP
- KaHyPar
- Zoltan/PaToH
- scikit learn