Fluid Communities: конкурентоспособный, масштабируемый

алгоритм обнаружения кластеров

Мы представляем алгоритм обнаружения сообществ (Fluid Communities), основанный на идее взаимодействия жидкостей в среде, расширения и сжатия в результате этого взаимодействия. Fluid Communities основан на методологии распространения, которая представляет собой самое современное решение с точки зрения вычислительных затрат и масштабируемости. Будучи очень эффективным, Fluid Communities может находить сообщества на синтетических графиках с точностью, близкой к лучшим текущим альтернативам. Кроме того, Fluid Communities - это первый алгоритм, основанный на распространении информации, способный определять переменное количество сообществ в сети.

1. Введение

Обнаружение сообщества (CD) - извлечение структурной информации из «сырой» сети. Сообщества обычно определяются наборами плотно взаимосвязанных вершин, которые слабо связаны с остальными вершинами графа. Поиск сообществ внутри графа помогает понять внутреннюю организацию графа, а также может использоваться для характеристики составляющих его сущностей (например, групп людей с общими интересами, продуктов с общими свойствами и т.д.). Одним из первых алгоритмов CD, предложенных в литературе, является алгоритм распространения меток (LPA). Хотя было показано, что другие алгоритмы CD превосходят его, LPA остается актуальным благодаря своей масштабируемости (с линейной вычислительной сложностью O(E)). В этой статье мы предлагаем новый алгоритм CD: алгоритм Fluid Communities (FluidC), который также реализует методологию эффективного распространения. Этот алгоритм имитирует поведение нескольких жидкостей (сообществ), расширяющихся и толкающих друг друга в общей, закрытой и неоднородной среде (в графе), пока не будет найдено равновесие. Инициализируя другое количество жидкостей в среде, FluidC может найти любое количество сообществ на графе. Насколько нам известно, FluidC является первым основанным на распространении алгоритмом с этим свойством.

1. Связанные работы

Самая последняя оценка и сравнение алгоритмов CD были выполнены Янгом и др., где следующие восемь алгоритмов сравнивались с точки зрения нормированной взаимной информации (NMI) и времени вычислений: : Edge Betweenness, Fast greedy, Infomap, Label Propagation, Leading Eigenvector, Multilevel (Louvain), Spinglass and Walktrap. Производительность этих восьми алгоритмов измерялась на искусственно созданных графиках, предоставленных тестом LFR, который определяет более реалистичные настройки, чем альтернативный тест GN. Один из основных выводов этого исследования заключается в том, что многоуровневый алгоритм является наиболее конкурентоспособным в целом с точки зрения качества. О подобном сравнении алгоритмов ранее сообщили Ланчинетти и Фортунато. В этом исследовании алгоритмы сравнивались в тестах GN, LFR и на случайных графах. В своем резюме авторы рекомендуют использовать различные алгоритмы при изучении структуры сообщества графа для получения информация, не зависящая от алгоритмов. Результаты показывают, что самый быстрый алгоритм CD - это хорошо известный алгоритм LPA, благодаря эффективности и масштабируемости методологии распространения.

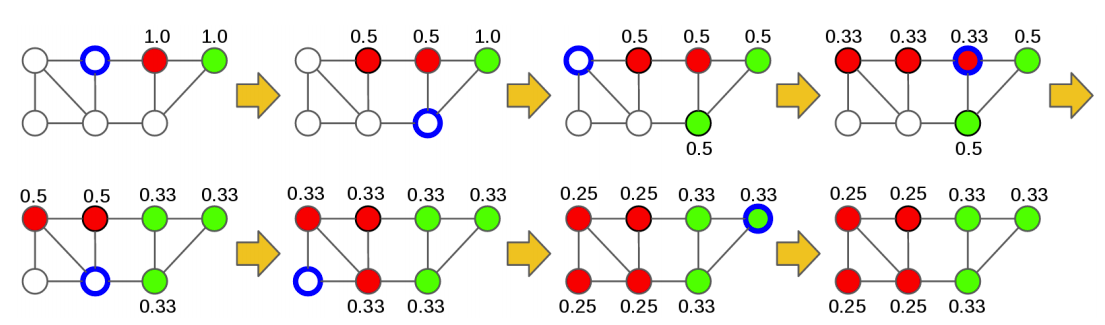


Рисунок 1 Рабочий поток FluidC для k = 2 сообществ (красный и зеленый). Каждая вершина, назначенная сообществу, помечена

1. Алгоритм Fluid Communities.

Алгоритм Fluid Communities (FluidC) - это алгоритм CD, основанный на идее введения ряда жидкостей (т.е. сообществ) в неоднородную среду (т.е. неполный граф), где жидкости будут расширяться и толкать друг друга, в зависимости от топологии среды, пока не будет достигнуто стабильное состояние.

В графе , состоящем из набора вершин и набора ребер , FluidC инициализирует жидкостных сообщества , где , Каждое сообщество инициализируется в другой случайной вершине . Каждый инициал этого сообщества есть связанная плотность d в диапазоне (0,1]. Плотность сообщества является обратной величиной числа вершин, составляющих указанное сообщество:

Обратите внимание, что жидкое сообщество, состоящее из одной вершины (например, каждое сообщество при инициализации), имеет максимально возможную плотность (d = 1.0).

FluidC работает через супершаги. На каждом супершаге алгоритм перебирает все вершины V в случайном порядке, обновляя сообщество, к которому принадлежит каждая вершина, используя правило обновления. Когда присвоенная плотность у вершин не меняется на двух последовательных супершагах, алгоритм сходится и завершается.

Правило обновления для конкретной вершины v возвращает сообщество или сообщества с максимальной агрегированной плотностью внутри эго-сети v. Правило обновления формально определяется в уравнениях 2 и 3.

где обновляется вершина, - это набор кандидатов в новые сообщества , - соседи , плотность сообщества , - это вершина принадлежащая сообществу и - символ Кронекера.

Обратите внимание, что может содержать несколько кандидатов от сообщества, каждый из которых имеет одинаковую максимальную сумму. Если содержит текущее сообщество вершины , не меняет свое сообщество. Однако, если не содержит текущего сообщества как новое сообщество из , правило обновления выбирает случайное сообщество внутри в качестве нового сообщества .

На этом формализация правила обновления завершена:

где - сообщества-кандидаты из уравнения 2 и - это случайная выборка из дискретного равномерного распределения .

Уравнение 4 гарантирует, что ни одно сообщество никогда не будет исключено из графа до тех пор, пока сообщество не сожмется в единственную вершину , имеет максимально возможную плотность по правилу обновления ( т.е. 1.0) гарантируя , и поэтому . Пример поведения алгоритма FluidC показан на рисунке 1.

3.1 Свойства

FluidC асинхронный алгоритм, где каждое обновление вершины вычисляется с использованием последнего частичного состояния графа (некоторые вершины могли обновить свою метку на текущем супершаге, а некоторые - нет). Последовательная синхронная версия FluidC (т.е. та, в которой все правила обновления вершин вычисляются с использованием окончательного состояния предыдущего супершага) не гарантирует, что плотности всегда будут согласовываться с Уравнением 1 (например, сообщество может проиграть вершину, но его плотность не может быть увеличена сразу). Следовательно, сообщество может потерять все свои вершины и быть удаленным из графа.

FluidC позволяет определить количество сообществ, которые необходимо найти, просто инициализируя другое количество жидкостей в графе. Это желаемое свойство для анализа данных, поскольку оно позволяет изучать граф и его сущности на нескольких уровнях детализации. Хотя несколько алгоритмов CD уже имели эту функцию (например, Walktrap, Fastgreedy), ни один из них не был основан на эффективном методе распространения.

Еще одна интересная особенность FluidC заключается в том, что он избегает создания сообществ монстров непараметрическим способом. Из-за разброса плотности между вершинами, составляющими сообщество, большое сообщество (по сравнению с остальными сообществами в графе) сможет сохранять свой размер и расширяться только за счет благоприятной топологии (т.е. наличия большого количества ребер внутри сообщества, которые компенсируют его более низкую плотность). На рисунке 2 показаны два случая такого поведения: один, когда большое сообщество способно защитить себя от внешних атак, и второй, когда это не так.

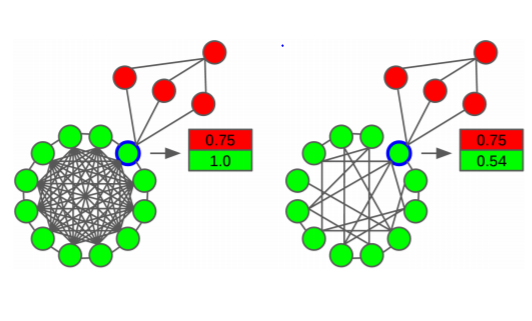
FluidC разработан для связанных, неориентированных, невзвешенных графов, и варианты FluidC для ориентированных и / или взвешенных графов остаются в будущем. Однако FluidC можно легко применить к несвязному графу просто выполняя независимо FluidC для каждого связанного компонента и складывая результаты.

Рис. 2 Два случая обновления правила на вершине (выделено синим). Левый случай показывает плотно связанное зеленое сообщество, которое может успешно защищать вершину. В правом случае зеленое сообщество более разрежено и потеряет выделенную вершину в пользу красного сообщества. Цифры соответствуют суммам из уравнения 2.

1. Оценка

Для оценки производительности мы используем тест LFR [6], измеряющий NMI, полученный на наборе графов с шестью различными размерами ( = 233, 482, 1000, 3583, 8916 и 22186) и 25 различных параметров смешивания ( µ от 0,03 до 0,75). Параметр смешивания - это средняя доля ребер вершин, которые соединяются с вершинами из других сообществ [6].

Чтобы гарантировать согласованность, было сгенерировано 20 различных графов для каждой комбинации размера графа и параметра смешивания. В результате получается 3000 различных графов (6 × 25 × 20), и та же стратегия оценки, которая использовалась в [16]. Помимо размера графа и параметра смешивания, тесту LFR также требуется список гиперпараметров для создания графа. Мы используем те же, что определены в [16]; максимальная степень , максимальный размер сообщества , средняя степень - 20, показатель степени распределения - 2 и показатель распределения размера сообщества - 1.

Эти эксперименты проводились с пятью наиболее конкурентоспособными алгоритмами CD, оцененными в [16] (Fastgreedy, Infomap, Label Propagation, Multilevel и Walktrap) и FluidC. Мы сообщаем результаты на Рисунке 3, где каждая панель содержит два графика. Нижний показывает среднюю производительность в NMI на 20 графах, построенных для различных значений µ (показано на горизонтальной оси), а верхняя показывает соответствующее стандартное отклонение (Std). Каждая линия графика представляет результаты алгоритма на графе разного размера (см. легенду панели). Среди всех вариантов нормализации метрики взаимной информации используется геометрическая нормализация, т.е. деление на квадратный корень из обеих энтропий.

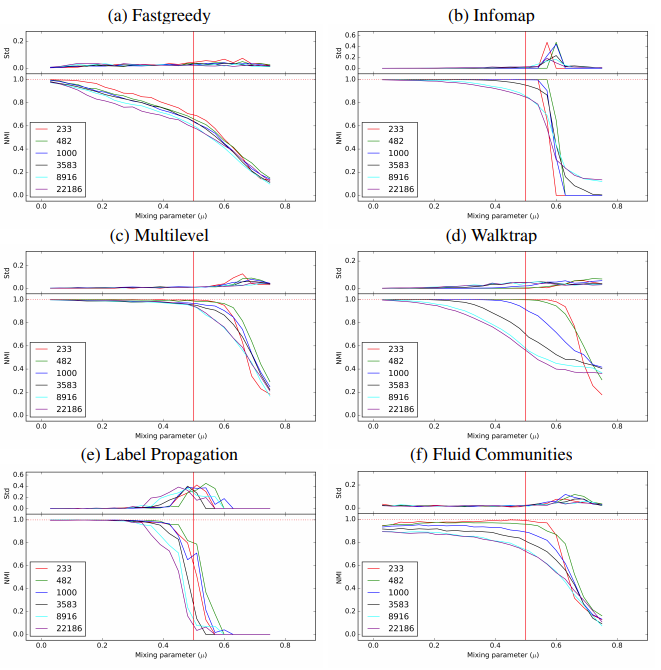
Прежде чем анализировать результаты, давайте проясним два аспекта процесса оценки. Во-первых, тест LFR может генерировать непересекающиеся графы. В этом случае независимое выполнение FluidC вычисляется для каждого подключенного подграфа отдельно, а сообщества, обнаруженные в разных подграфах, добавляются для измерения общего NMI. Во-вторых, FluidC требует указать количество сообществ, которые нужно найти: k, что является неизвестным параметром. В целях сопоставимости мы сообщаем результаты, полученные с использованием k с высочайшей модульностью. Это аналогично другим алгоритмам, которые также требуют задавать k (например, Fastgreedy и Walktrap).

Рис. 3 Производительность NMI шести алгоритмов CD. Каждая панель разделена на два графика, нижний показывает среднюю производительность NMI по 20 случайным графам, созданным с одинаковыми свойствами, включая размер и параметр смешивания. На верхнем графике показано стандартное отклонение. Разные построенные линии соответствуют разным размерам графов. Для справки, каждая панель содержит две линии, одна вертикальная для обозначения параметра смешивания 0,5 ( µ) и одна пунктирная горизонтальная линия для обозначения идеальной оценки NMI (NMI = 1,0). Представленные здесь характеристики Walktrap существенно отличаются от Yang из [16] из-за ошибки в их экспериментах, признанной авторами.

В наших экспериментах Multilevel достиг лучших результатов NMI на большинстве сгенерированных графов, в то время как Fastgreedy и LPA явно уступали остальным алгоритмам. Остальные три алгоритма, Walktrap, Infomap и FluidC, были конкурентоспособными и показали результаты, близкие к Multilevel. В контексте Walktrap и Infomap FluidC ведет себя довольно специфично. Лучше на графах больших размеров, чем Walktrap; для наибольшего вычисленного графа (), FluidC превосходит Walktrap для всех µ в диапазоне [0,33,0,66]. FluidC также более устойчив к большим параметрам смешивания, чем Infomap, который не может обнаруживать соответствующие сообщества для µ> 0,55. Производительность FluidC немного ниже оптимальной (NMI от 0,9 до 0,95) при низких параметрах смешивания (µ ≤ 0,4). Это связано с тем, что сообщества, созданные в графе с низкими параметрами смешивания, очень плотно связаны и имеют лишь несколько ребер, соединяющих их с другими сообществами, ребра, которые действуют как узкие места. Эти узкие места могут иногда препятствовать нормальному потоку сообществ в FluidC, что приводит к неоптимальным результатам.

На практике неоптимальная производительность FluidC на графах с очень низким параметром смешения является незначительным неудобством. Графики реального мира часто бывают большими и имеют относительно высокие параметры микширования, для которых FluidC особенно подходит. Напротив, рекомендуемый алгоритм для обработки графов с низкими параметрами смешивания будет LPA, поскольку он дает оптимальный результат в этих случаях, быстрее и лучше масштабируется, чем альтернативы.

1. Масштабируемость

Основная цель алгоритма FluidC - предоставить сообщества высокого качества с возможностью масштабирования, чтобы сообщества хорошего качества также можно было получить из крупномасштабных графов. В предыдущем разделе мы увидели, как производительность FluidC с точки зрения NMI сравнима с лучшими современными алгоритмами (например, Multilevel, Walktrap и Infomap). Затем мы оцениваем масштабируемость FluidC, чтобы показать ее актуальность в контексте больших сетей.

Чтобы проанализировать вычислительные затраты FluidC, мы сначала сравниваем стоимость одного полного супершага (проверка и обновление сообществ всех вершин в графе) со стоимостью LPA. LPA - это самый быстрый и более масштабируемый алгоритм из современных [16], поэтому мы используем его в качестве основы для масштабируемости в этом разделе. На рисунке 4 показано среднее время на итерацию с использованием того же типа графов, который использовался при оценке NMI. Результаты показывают, что время вычислений на итерацию FluidC практически идентично времени вычислений LPA для всех размеров графов и параметров смешивания. Существенно, что оба алгоритма практически не зависят от изменения параметра микширования.

Помимо стоимости одного супершага, мы также исследуем общее количество супершагов, необходимых для сходимости алгоритма. На рисунке 5 показана эта информация как для FluidC, так и для LPA. Для этого эксперимента мы устанавливаем параметр FluidC k на реальное значение. Это сравнение актуально для параметров смешивания до 0,5. При превышении этого значения LPA создает единое гипер-сообщество после трех супершагов (NMI = 0.0, см. Рисунок 3, панель e). Тем не менее, количество супершагов, необходимых FluidC для схождения, никогда не превышает 13.

Эти результаты показывают, что FluidC и LPA похожи как по времени на супершаг, так и по количеству супершагов, что означает, что оба алгоритма аналогичны с точки зрения вычислительных затрат (с линейной сложностью O (E)) и масштабируемости. Чтобы получить дополнительные доказательства в этом отношении, а также оценить масштабируемость наиболее подходящих альтернатив, теперь мы рассмотрим оценку более крупных графов. Мы генерируем графы с 60 000 и 150 000 вершин по той же методологии, которая описана в § 4, и измеряем время вычисления LPA, FluidC, Multilevel и Walktrap (три самых быстрых алгоритма согласно [16] плюс FluidC). На рисунке 4 показана масштабируемость каждого алгоритма, где непрерывные линии на заднем плане соответствуют различным параметрам смешивания (от 0,03 до 0,75), а большая пунктирная линия с маркерами указывает вычисленное среднее значение по всем 25 параметрам смешивания.

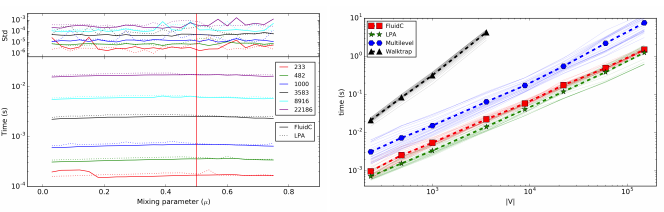


Рис. 4 Слева: время вычисления на итерацию для алгоритмов FluidC и LPA при обработке графов различного размера и параметра смешивания. Справа: масштабируемость FluidC, LPA, Multilevel и Walktrap на графах до 150 000 вершин. Пунктирная линия показывает среднее значение для различных значений параметров смешивания (от 0,03 до 0,75).

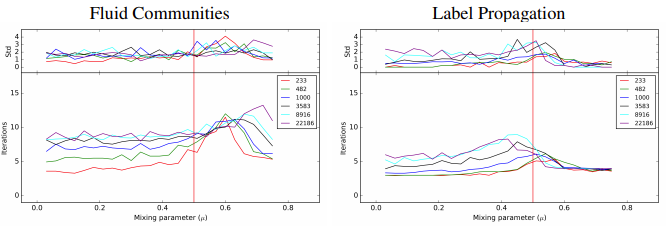


Рис. 5 Количество итераций до сходимости для алгоритмов FluidC и LPA при обработке графов различного размера и параметра смешивания.

Согласно результатам, показанным на рисунке 4, LPA имеет самое низкое время вычислений, за ним следует FluidC. Однако результаты LPA ошибочно оптимистичны, поскольку алгоритм особенно быстр для больших параметров микширования, когда он получает нулевой NMI после выполнения только трех супершагов (см. Панель e на рисунке 3 и рисунок 5). Если это принять во внимание, LPA и FluidC обладают аналогичной масштабируемостью.

Walktrap значительно медленнее, чем остальные, и результаты для графиков больше 3000 вершин не показаны. Multilevel в 5 раз медленнее, чем LPA / FluidC, и его стоимость растет быстрее. В то время как наклон LPA / FluidC, вычисленный с помощью линейной регрессии, составляет примерно 10- 6, уклон Multilevel близок к 10- 5. Обработка крупномасштабного графа, такого как PageGraph [7] (| V | = 3500M вершин) для FluidC потребуется примерно 9 часов, а для Multilevel - более двух дней (примерно 49 часов).