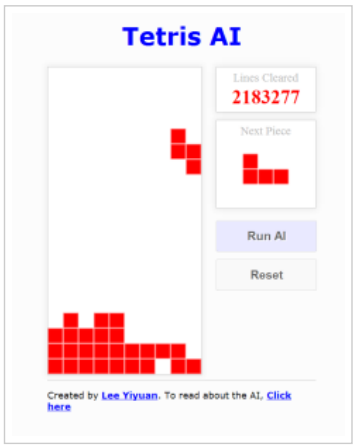
Tetris AI - The (Near) Perfect Bot

Йиюн Ли / 14 апреля 2013

В этом проекте, мы разрабатываем AI (онлайн демо здесь) , которые могут бесконечно четкие линии в одной игре Тетрис (я должен был остановить его , потому что мой Iapry был выжигание).



AI через две недели

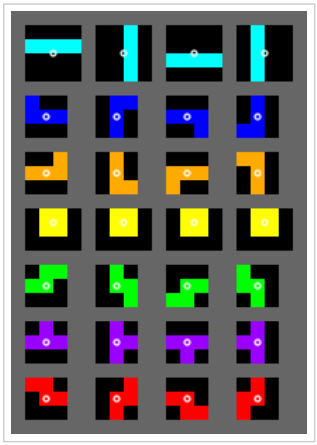
Определение игры

Пожалуй, самое интересное наблюдение я сделал в ходе разработки ИИ был таков: несмотря на существование «стандартного» Тетрис директивы , есть еще много вариантов игры. Если вы должны были играть в тетрис онлайн, шансы на то, что это не будет «стандарт» один.

Для этого AI, я застрял в «официальные» директивы настолько близко, насколько это возможно. Ради уменьшения неопределенности, вот правила я прилипшие к:

1. Сетки Тетрис 10 клеток в ширину и 22 клетки высотой, с 2 верхних строк скрыты.
2. Все тетромины (Тетрис штуки) начнутся в середине 2 верхних строк.
3. Есть 7 тетромино: «Я», «O», «J», «L», «S», «Z», «T».
4. Вращение системы Супер используется для всех вращений.
5. «7 система» генератор случайных чисел используются для рандомизации следующих частей.
6. Один опережение куска разрешено (игрок знает, что следующая часть будет).

Ниже приведена таблица всех возможных частей в соответствии с этими правилами.



Возможные части и вращение

Лучший возможный шаг

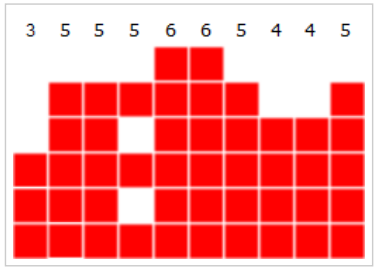
Наша цель состоит в том, чтобы очистить как можно больше линий, как это возможно, и, следовательно, сделать так много ходов, как это возможно.

Для достижения этой цели, наш ИИ будет выбрать наилучший ход для данной Tetris кусок, пробуя все возможные ходы (вращение и положение). Он вычисляет балл для каждого возможного перехода (вместе с опережением частью), и выбирает один с лучшим счетом, его следующим шагом.

. Счет для каждого шага вычисляется путем оценки сетки шаг мог бы привести Эта оценка основана на четырех эвристики: **совокупный рост** , **комплектных линий** , **отверстий** и **тряски** , каждый из которых ИИ будет пытаться либо минимизировать или максимизировать.

Совокупная Высота

Эта эвристика говорит нам, как «высокий» сетка есть. Для того, чтобы вычислить суммарную высоту, мы возьмем сумму высоты каждой колонки (расстояние от самой высокой плитки в каждом столбце в нижней части сетки).

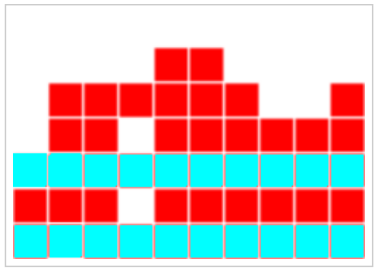


Совокупная высота = 48

Мы хотим , чтобы **минимизировать** это значение, так как нижняя совокупная высота означает , что мы можем упасть более штуки в сетку до попадания в верхней части сетки.

Комплектные линии

Это, вероятно, наиболее интуитивный эвристический среди четырех. Это просто число полных строк в сетке.

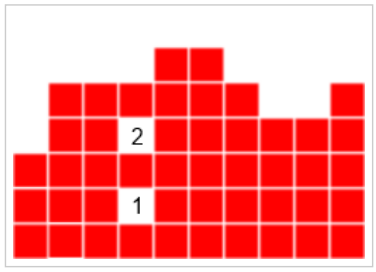


Линии = 2

Мы хотим , чтобы **максимизировать** количество полных линий, поскольку очистка линий является целью ИИ, и очищающие линии дадут нам больше места для более штук.

Отверстия

Отверстие определяется как пустое пространство таким образом, что существует по крайней мере одна плитка в той же самой колонке над ним.

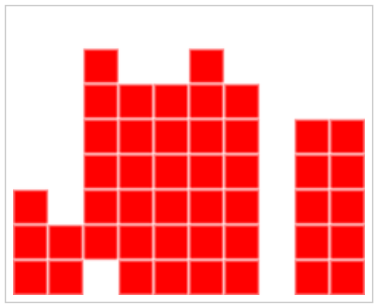


Количество отверстий = 2

Отверстие труднее очистить, потому что мы должны очистить все линии над ним , прежде чем мы сможем достичь отверстия и заполнить его. Таким образом , мы должны **минимизировать** эти отверстия.

болтанки

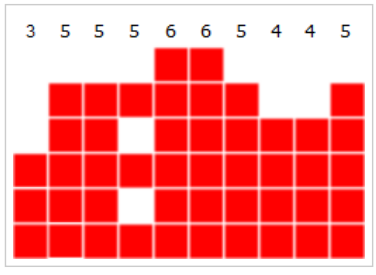
Рассмотрим случай, когда мы получаем глубокий «хорошо» в нашей сетке, что делает его нежелательным:



Неидеальный сетки

Присутствие этих скважин показывает, что линии, которые могут быть очищены легко не удаляются. Если же должны были быть покрыты, все строки которой также пролеты будет трудно очистить. Обобщая идею «хорошо», мы определяем эвристики, которые я буду называть «болтанка».

Болтанка сетки говорит нам изменение его высоты колонны. Он рассчитывается путем суммирования абсолютных различий между всеми двумя соседними столбцами.



Ухабистость = 6

В приведенном выше примере,

$$bumpiness = 6 = |3-5| + |5-5| + ... + |4-4| + |4-5|$$

Для того, чтобы убедиться , что в верхней части сетки монотонный , как это возможно, ИИ будет пытаться **минимизировать** это значение.

Ввод эвристики вместе

Вычислит теперь оценку сетки, взяв линейную комбинацию четыре нашей эвристики. Она задается следующей функцией оценки:

$$a \times (AggregateHeight) + b \times (CompleteLines) + c \times (Holes) + d \times (Bumpiness)$$

где a, b, c, dпостоянный paramters.

Мы хотим , чтобы минимизировать **суммарную высоту** , **отверстие** и **болтанку** , поэтому мы можем ожидать , a, c, dчтобы быть отрицательными. Кроме того , мы хотим , чтобы максимизировать количество **комплектных линий** , таким образом , мы можем ожидать , Bчтобы быть положительным.

Я использовал генетический алгоритм (GA) (объяснено в полной подробно ниже), чтобы произвести следующий оптимальный набор параметров:

$$\begin{aligned} a &= -0.510066 \\ b &= 0.760666 \\ c &= -0.35663 \\ d &= -0.184483 \end{aligned}$$

С помощью этого набора параметров и оценки формулы, ИИ может выбрать наилучший ход по исчерпав все возможные хода (в том числе опережения части) и выбрать один с наибольшим количеством очков.

Генетический алгоритм

Наборы параметров, 4D векторы и блок 3-сфера

Каждый возможный набор параметров (A, B, C, D) может быть представлен в виде вектора в $\mathbb{R}^4$,

$$\vec{p} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

Стандартный генетический алгоритм для вещественных генов будет включать поиск все пространства решений ($\mathbb{R}^4$) для оптимального набора параметров. В этом проекте, однако, мы должны рассматривать только точки на единичной 3-мерной сферы (то есть 4-мерного единичного шара). Это потому , что функция оценки определена выше линейный функционал и результаты сравнения инвариантны относительно масштабирования (функций баллов).

Чтобы понять это с более математической точки зрения, первым переписать функцию оценки в контексте векторов, как

$$f(\vec{x}) = \vec{p} \cdot \vec{x}$$

где

$$\vec{p} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}, \vec{x} = \begin{pmatrix} \text{Aggregate Height} \\ \text{Complete Lines} \\ \text{Holes} \\ \text{Bumpiness} \end{pmatrix}$$

Предположим , что мы хотим сравнить между двумя движениями, перемещать 1 и переместить 2, чтобы решить , какой шаг является «лучше» (т.е. дает более высокий балл). Предположим также , что ход 1 и 2 производит значения $\vec{x_1}$и $\vec{x_2}$ соответственно. Для того, чтобы проверить , если шаг 1 лучше , чем шаг 2, мы должны проверить , если $f(\vec{x_1}) > f(\vec{x_2}) \iff f(\vec{x_1}) - f(\vec{x_2}) > 0$. Если это условие не выполняется , то это тривиально сделать вывод , что движение 2 лучше или одинаково хорошо по сравнению с 1 двигаться.

Теперь,

$$f(\vec{x_1}) - f(\vec{x_2}) = \vec{p} \cdot \vec{x_1} - \vec{p} \cdot \vec{x_2} = \vec{p} \cdot (\vec{x_1} - \vec{x_2})$$

Предположим , что $\vec{p} \cdot (\vec{x_1} - \vec{x_2})$это положительный или отрицательный. Лучше двигаться затем будет двигаться 1 и переместить 2 соответственно. Здесь мы можем увидеть , как результат сравнения инвариантен относительно масштабирования параметров - мы можем умножить $\vec{p}$на любом положительном вещественном (скалярном) постоянная и **знак** вышеуказанной разницы останутся прежним (хотя **величина** отличается - но это не имеет значения) , в этом случае решение не отличается!

Все векторы параметров в том же направлении, эквивалентные результаты. Таким образом, достаточно рассмотреть только один вектор для каждого направления. Это оправдывает ограничение поиска только точки на поверхности единичной 3-сферы, как поверхность сферы охватывает все направления возможно.

Теперь мы будем двигаться дальше, чтобы определить процедуру пригодности функции, процедуры выбора, оператор кроссовера, мутации и рекомбинации оператор используется для настройки ИИ.

Фитнес-функции

Население первым инициализируется 1000 векторов параметров блока.

Для каждого вектора параметров, мы запускаем AI для 100 игр, каждая из которых не более 500 Tetromino штук (т.е. 500 в большинстве ходов). Мы выпускаем приспособленность $fitness(\vec{p})$любого вектора один параметр $\vec{p}$будет равен общему числу линий расчитана. Наихудшие приспособленности 0, в этом случае AI не очищают никаких линий для всех игр.

выбор турнира

Родительские особи выбираются для воспроизведения с помощью отборочного турнира. Мы выбираем 10% (= 100) населения в случайном порядке, и два сильнейшие люди в этом подпул исходить для кроссовера, чтобы произвести новое потомство. Этот процесс повторяется до тех пор число новых потомков, произведенных не достигнет 30% от размера популяции (= 300).

Средневзвешенное кроссовер

Для кроссовера, два родительских векторы $\vec{p_1}, \vec{p_2}$объединяются с использованием векторной формы средневзвешенных. Единичный вектор потомства $\hat{\vec{p'}}$может быть произведен , где

$$\vec{p'} = \vec{p_1} \cdot fitness(\vec{p_1}) + \vec{p_2} \cdot fitness(\vec{p_2})$$

По сути, вектор потомства лежит между двумя родительскими векторами на единичной 3-сфере, но склоняется к слесарю родителей на сумму, которая возрастает с увеличением разницы между приспособленностью два родителей. Это отражает фаворитизм оператора кроссовера к электромонтера родителя.

оператор Мутация

Каждый вновь произвел потомство получает небольшой шанс (5%) мутировать. Если это действительно мутировать, то случайная составляющая вектора потомка корректируется на случайную величину до ± 0.2. Вектор затем нормализуется.

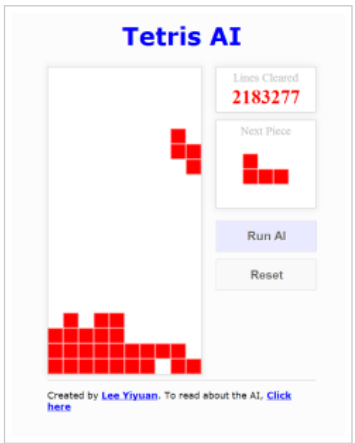
Как мы работаем с единичными сферами, я предпочел бы вращение векторов потомков на определенный угол. Однако, в отличие от вращений в 3D, 4D повороты намного больше озадачивает. Таким образом, я выбрал для вышеупомянутого более простого оператора мутации.

Удалить-н-последняя замена

После того, как количество производимых потомков достигает 30% (= 300) от исходного размера популяции, самые слабые 30% особи в популяции будут удалены и заменены вновь произведенными потомками, образуя следующее поколение населения. Численность населения остается тем же (1000). Затем можно повторить весь процесс, пока население не будет достаточно хорошей форме, после чего алгоритм завершается.

Результаты

AI был реализован в JavaScript , и протестирован с Google Chrome (он также работает для большинства других браузеров). Вы можете попробовать его в Интернете здесь . Исходный код для реализации можно найти здесь . AI еще работает примерно через две недели - это расчистили 2,183,277 линии к тому времени.



AI через две недели

Вывод

Через ходе этого проекта я попробовал некоторые другие эвристики, такие как самая высокая высота колонны и количество последовательных отверстий.

Оказывается, что некоторые из них были лишними, и некоторые из них были заменены лучшими эвристики (например, самая высокая высота колонны эвристический была заменена совокупной высоты), так как это дает лучшие результаты.

Я хотел бы также подчеркнуть , что ИИ был настроен для **определенного набора правил** (я определил их в первых разделах этой статьи). Результаты могут или не могут быть одинаковыми для другого набора правил.

Например, используя наивный кусок генератора случайного вместо этого может приводить к неприличию длинной последовательности «S» или «Z» плитка, которые будут увеличивать сложность ИИ. Позволить больше lookaheads также позволит ИИ создавать более сложные движения, в этом случае он будет, вероятно, работают лучше (если правильно настроен).

Таким образом, при сравнении между результатами различных Tetris AIs, большое внимание следует уделить набору правил, используемых для справедливого сравнения, которое будет сделано.

14 апреля 2013 в проектах . Ключевые слова: ай , эвристический , тетрис

Похожие сообщения


Решение «Rotation» головоломки поэтапно


2048 AI - Интеллектуальный Bot


Решение судок по возврату

← Визуальная Лагранж Интерполяция

Native Code VS управляемого кода - Производительность и развитие →

18 мыслей на « Тетрис AI - The (Near) Perfect Bot »


DPR 21 августа 2013 в 10:17

«Кажется , что количество линий очищены никогда не может превышать 20% от числа тетраминых помещенным.»
Это легко объяснит =) Каждая строку будет принимать 20 блоков , если ширина доски 20. Поскольку каждый Tetromino 4 блоков, это займет ровно 5 тетрамино , чтобы очистить линию. Таким образом, теоретический предел в том , что 20% от числа тетрамина будет составить очищается линия. На практике это число будет ниже , так как там будет тетрамиными , что вы разместили и не были очищены , когда вы проигрываете. (Например, если вы очистили 1000 строк с 5000 блоков , но затем потерял с следующих 200 блоков.) Но чем

дольше вы выжить, тем ближе вы к этому проценту. Я заметил то же самое , когда я программирование Тетриса AI в последнее время , хотя я использую другую ширину сетки , так это был другой процент.

Из любопытства, что линии средних освобоженные для ИИ на нормальный размере сетки? Лучшее , что я до сих пор в среднем ~ 16k линии расчищенной из 1000 игр. Но я не испытывал болтанку и засоры в рамках скоринга , как вы упомянули здесь. Кроме того , я не использовал генетический алгоритм; Я осуществил то , что я называю нарезка из- за отсутствия лучшего слова. Это место , где я могу проверить 100 различных значений на первый признак, сохранить лучшее, а затем сделать тест на 100 значений для второго признака, сохраняя при этом первый признак, и т.д. Выполнение этого требует принятия нескольких сумасшедших предположений о чертах линейно независимы, но я обнаружил , что тесты , чтобы быть проще (по крайней мере для меня) , и результатов , чтобы быть приемлемыми.
Кроме того , вы позволили ИИ выбрать , где блок будет стартовать в направлении слева направо , или же вы его начать в середине , как с оригинальным тетрис? Я нашел , что это делает несправедливое, существенное различие в возможностях искусственного интеллекта, так что мне пришлось отключить его в состоянии иметь начало куска в любом месте вдоль верхней части. Первоначально я сделал это , потому что я был ленив и не хотел , чтобы имитировать движение влево-вправо вдоль верхней, так и вниз. ^^

Ответить

Ли Yiyuan 21 августа 2013 в 11:12 вечера

Привет DPR,

Это очень хорошее объяснение 20%, возможно , мы могли бы установить более общую связь , которая обеспечивает ограничение для определенной ширины.
Я не тестировал для нормального размера сетки, но я буду обновлять ее в будущем , так что размер сетки может быть установлен. Изначально я ушел из болтанку , а также (слишком много параметров является боль , когда дело доходит до генетического алгоритма), но когда я попробовал, ИИ , казалось, работать лучше.
Что касается левого и правого преимущества, Нео, я не сделал это начинать с середины , а также (потому что я тоже был ленивым XD). Но вы правы, это довольно несправедливо. Я буду смотреть на изменение поведения ИИ , так что тетрис блоки приходят вниз от центра.

Еще раз спасибо за комментарий и держать их прибытием! ^^
-Yiyuan

Ответить

Ли Yiyuan 25 мая 2014 в 5:58 вечера

Привет еще раз DPR!

С тех пор я обновил правило к одному указанным в «Официальном» Руководстве Tetris (http://tetris.wikia.com/wiki/Tetris_Guideline). Я также переместил AI онлайн как приложение Javascript - это хлопот бесплатно! Как проверить результаты.

Благодаря!
-Yiyuan

Ответить

DPR 27 ноября 2014 в 4:49

Это выглядит очень красиво =) В демонстрационных целях, Javascript приложение работает очень хорошо на что делает его легче для людей, чтобы проверить. Я посмотрел через источник Javascript. Это было довольно интересно увидеть места, где вы реализовали некоторые вещи, подобно мне, и другие места, где вещи были сделаны по-разному. Там редко один правильный способ сделать это, но некоторые подходы, кажется, настолько хорошо, что люди, естественно, стремятся к их использованию. Использование для цикла, чтобы проверить все оптимальные результаты различных вращений, к примеру.

Pingback: 2048 AI - Интеллектуальный Bot | Код My Дорога

Тони 7 марта 2015 в 11:35

Это действительно хорошо. Я обычно выключаю аю и беспорядок его немного и повернуть его назад, чтобы увидеть, насколько хорошо он может исправить это. Оказался очень хорошо не может добраться до ума, но да. Теперь то, что я хочу сказать, что вместо того, чтобы его ускорить его, нажав на кнопку вниз для того, чтобы подтолкнуть пространство вместо этого для того, чтобы иметь возможность поместить блок очень быстро.

Ответить

bengoldberg 2 апреля 2016 в 12:01

Что делать, если выбор кусок не был случайным, но вместо тетромино были выбраны выбрали другой AI, одна чья «цель», чтобы заставить игрока потерять?

Ответить

Ли Yiyuan 2 апреля 2016 в 5:07 вечера

Это хороший вопрос. В этом случае мы можем перепроектировать ИИ рассматривать игру как игра двух игроков в и использовать Минимакс вместо того, чтобы вычислить оценку каждого решения для каждого хода.

Это будет очень похоже на забегая вперед, за исключением того, что ИИ теперь предполагает, что в ближайшие несколько штук являются частями, которые минимизируют свои будущие общие оценки.

Ответить

Умар Хан 27 июля 2016 в 12:22 AM

Я считаю, что в стандартном тетрис достаточно длинную последовательность S, Z частей неизбежно приводит к потере поэтому второй AI всегда будет выигрывать при достижении этого предела.

Митч 12 апреля 2016 в 11:34 вечера

Какой хороший кусок работы ☺
Но все же есть одна вещь , которая заставляет меня немного запутался: Не должно ли быть разница между рейтингом сетки с большим отверстием и сетки с небольшим отверстием?
Как это:

XXXX
XXXX
XX_X
XXXX

и это:

XXXX
XX_X
XX_X
XXXX

(Где X представляет собой плитки и _ дыра)

Кажется, что эти две сетки получить тот же счет сетки. Should't быть второй сетки менее желательны?

Спасибо,
Митч

Ответить

Ли Yiyuan 13 апреля 2016 года в 12:00 AM

Да, в самом деле. Любое пустое пространство, по крайней мере , одной плитки над ней в том же столбце, определяется как «дыра». Это может быть менее интуитивным и более аксиомой *действительно* сделать вывод из этого определения , что во втором случае вы изложили, две «дыры» s существует вместо того , чтобы только один ☺

Ответить

Митч 13 апреля 2016 в 6:59 вечера

Понимаю.
Тем не менее, есть еще один вопрос , я невежественный с, может быть , у вас есть мнение по этому вопросу :

Вы писали: «Счет для каждого шага вычисляется путем оценки сетки шаг мог бы привести к».

Я думаю , что движение не заканчивается , когда кусок места, но когда дополнительно все полные линии очищаются.
Потому что (например) отверстия , которые появляются при размещении куска может disapper после того, как линия была очищена.

Вы вычислить счет сетки без очистки комплектных линий. Разве это не искажает счет сетки?

Йиюн Ли 13 апреля 2016 в 10:21 вечера

В самом деле, не очищая линии первых, этот конкретный компонент, похоже, совсем немного конфликта с тремя другими компонентами.

Однако, для этого проекта, я не ясно линии первого перед вычислением счет, потому что тогда я думал, что сразу же в результате сетка будет предоставить больше информации для ИИ работать с для получения лучших результатов.

Возможно, очистив линии первых перед вычислением счет сетки, «количество полных строк» компонент может фактически оказаться излишним. Это не возможно изменить AI таким образом, что она очищает линии первого перед вычислением счет сетки, и таким образом, что «число полных строк» компонент больше не включается в расчете партитуры сетки. При выполнении тестов на измененном ИИ можно определить, по крайней мере, ярмарки, а также текущую версию.

Митч 13 апреля 2016 в 11:26 вечера

Да, я тоже так думаю, очищая линию первой перед вычислением счет сетки> может <сделать компонент «numberOfCompleteLines» излишним, так как очистка линии делает совокупный рост стало намного меньше , что, конечно, желательно, делая хода, результат в расчищенной линии имеет более высокий балл сетки.
Я думаю , что самый интересный вопрос здесь: делает удаление результата «numberOfCompleteLines» в другом поведении бота? Может быть , лучший ход остается неизменным при расчете баллов сетки после очистки линии.

Крис 17 июля 2016 в 11:09 вечера

Привет Ли,

то , что удивительный проект! В настоящее время я пытаюсь понять генетические алгоритмы для решения совершенно аналогичные проблемы. То , что я не совсем понимаю , после прочтения вашей статьи, как вы

рассчитали параметры.
В коде JS эти параметры жёстко, так что я предполагаю , что вы имели какой - то другой код, вычисленные их?
Когда я их меняю ИИ все еще , кажется, работает, просто не так хорошо , по- видимому.
Мне было бы интересно посмотреть , как на самом деле их вычисления.

Следите за хорошую работу ☺

Ответить

Йиюн Ли 27 июля 2016 в 12:10

Привет Крис,

К сожалению, расчет параметров были некоторое время назад , и я больше не имею код доступен со мной.
Для полноты картины , я могу переписать код оптимизации снова когда - нибудь в ближайшее время в будущем после того, как я сделал с моим сроком призыва на военной службе.
Методы , используемые в процессе оптимизации были точно так , как описано в статье , и вы можете обратиться к ним для общей идеи , как сейчас.

Ура!

Ответить

Tizen E. 21 марта 2017 в 1:58 вечера

Привет Ли, На
самом деле удивительный проект , и я фанат Clannad тоже!

Ответить

Йиюн Ли 21 марта 2017 года в 2:00 вечера

:>

Ответить

оставьте ответ

Enter your comment here...

Сообщения по месяцам

Апрель 2015 (1)
Март 2015 (1)
Май 2014 (2)
Март 2014 (1)
Ноябрь 2013 (1)
Август 2013 (3)
Июль 2013 (1)
Июнь 2013 (1)
Апрель 2013 (2)