Комбинаторная оптимизация.
Практика 2.
Отчет

Кузнецов Дмитрий Сергеевич
БПМИ171 НИУ ВШЭ

Апрель 2020

1 Преамбула

Сразу заметим, что мое лучшее решение прошло только базовые тесты. Тем не менее были опробовано множество разных подходов, но, к сожалению, это не дало ощутимого эффекта. Есть подозрение, что была какая-то бага, которую не удалось обнаружить. Далее я попробую разобрать это поподробнее.

2 Менеджер доски

Перед тем как запускать какой-либо метод поиск, необходимо построить сущности для взаимодействия с состоянием доски.

Был реализован менеджер, который хранит в виде координат фигуры на доске и поддерживает актуальное число попарных коллизий. Получение числа коллизий происходит за 1 обращение ($O(1)$), поддержка числа попарных коллизий стоит ($O(1)$) при вставках новых фигур на доску и при перемещении с одной позиции на другую.

3 Базовое решение

В качестве базового решения был рассмотрен самое наивное представление локального поиска. В качестве функции соседа была положено состояние доски, когда какой-то ферзь переставлен вдоль своей строки, на другую колонку.

Постановка базового локального поиска - запускаем L локальных итераций, выбираем случайную фигуру (которую позволено переставлять), перебираем вдоль строки лучшую перестановку (с точки зрения попарных коллизий) производим перемещение и переходим к новой итерации. В базовой версии выполнено две точки останова - по числу итераций и по достижении решения с нулевым числом коллизий.

Данный подход легко справляется с первыми двумя тестами, но возникают проблемы уже при доске размера 15-20 (долго ищет и может с большой вероятностью не найти решение), т.к. застревает в локальном минимуме очень легко.

В качестве начального приближения полагалась случайное дополнение изначальной доски, у которой 0 коллизий вдоль каждой строки (данный инвариант поддерживает в процессе решения).

4 Еще локальный поиск

Далее было рассмотрено потенциальное улучшение метода. Давайте теперь в качестве функции соседа рассмотрим не перестановку вдоль строки, а чуть более хитрую вещь. Берем некоторую случайную пару фигур и меняем их столбцы местами, т.е. после этой операции должен произойти переход фигур $(x, y), (a, b)$ в фигуры с координатами $(x, b), (a, y)$ (Данный метод был подсмотрена в статье *A Polynomial Time Algorithm for the N-Queens Problem, Rok Sosic 1990*). Важно заметить, что с такой функцией соседей у нас не меняется множество столбцов и строк, которые заняты фигурами. А значит наши начальные инициализации обязаны расставить фигуры так, чтобы не было свободных строк и столбцов и как следствие коллизиций по строкам и столбцам (это останется инвариантом нашего подхода).

Заметим, что теперь мы выполняем шаг быстрее, за $O(1)$ против $O(N)$ в предыдущем подходе, т.к. мы выполняем случайный выбор пар фигур для перестановки.

В качетсве метода инициализации - ставим дополняющие фигуры в случайные строки и столбцы так, чтобы отсутствовали коллизии по строкам и столбцам (по факту просто случайная перестановка столбцов для свободных фигур).

Так же еще был опробован вариант не случайных выборов пар, а поиск фигуры, которая дает наибольшее число коллизий и подбор для нее пары для перестановки (пробовал делать это и случайным образом и максимизируя по числу коллизий). И так и так подход работал хуже, чем перестановки случайных пар.

5 Имитация отжига

После локального поиска был реализован отжиг. Базовая версия - обычный отжиг с температурой $t \sim 1, \alpha = 0.9$. Остальная конфигурация осталась без изменений: инициализация - фигуры без коллизий по строкам и столбцам, в качестве функции соседей перестановка случайной пары.

Заметим, что значение температуры должно быть пропорционально приращению числа попарных коллизий при перестановке случайной пары. Предполагается, что с ростом размера доски должно расти приращение засчет роста числа фигур. Если данная гипотеза подтверждается тогда мы можем улучшить метод подбором начального значения температуры на основе размера доски. Это бы улучшило сходимость отжига.

Для подтверждения гипотезы был проведен эксперимент: для каждого размера доски генерируем 1000 случайных досок (из пустой доски делаем случайную инициализацию нашим методом инициализации), для каждой такой доски делаем 100 случайных перестановок случайных пар и сохраняем приращение числа коллизий. В итоге получаем оценку (берем среднее замеров) методом Монте-Карло на мат. ожидание приращения числа коллизий для каждого размера доски (еще для уверенности будем считать выборочную дисперсию). Ожидалось, что с ростом размера доски будет расти и матож, но как оказалось на практике, матож приращения коллизий при любых размерах доски колеблется около 2.7. Не то, что мы ожидали, но тоже полезное знание. В качестве начального значения температуры давайте теперь положим 2.7.

Я так же попробовал руками брать в качестве начального значения температуры какие-либо линейные зависимости от размера доски, прироста качества это не дало.

6 Мета-эвристики

После имплементации и пробы предыдущих методов, сразу же был реализован и закреплен в конечном решении мета-метод: запуск из разных точек иниициализации всех ранее описанных методов.

Т.е. фиксируем один из методов поиска (лок. поиск, отжиг при тех или иных постановках) и $globalIter$ раз перезапускаем метод из разных начальных приближений. Это дает ощутимый результат, что и наконец позволило выбить 8 базовых тестов.

Экспериментально было выявлено, что это необходимое условие для демонстрации хороших результатов, т.к. и отжиг и локальный поиск эффективно застревает в локальных минимумах.

По этой причине следующая цель экспериментов была - разработка точек останова, которые бы определяли, что мы застряли в локальном минимуме, чтобы как можно скорее перезапустить метод из нового приближения.

Были рассмотрены следующие идеи:

1. Фиксируем нижнее значение температуры, при достижении которой мы останаливаем метод и перезапускаем на новой инициализации. Например 0.001. Данный подход только ухудшил результаты, т.к. очень часто не доводил ветки, на котором потенциально находились оптимальные решения.

2. Если K итераций значение функции не меняется (либо не уменьшается) выскакиваем из метода и делаем рестарт.

3. tabu-search (об этом далее)

В итоге было решено убрать ограничения на итерации (как глобальные, так и локальные) и ограничивать работу мета-поиска по времени (в 10с), а выходить из локальных поисков с помощью критериев останова. Так мы избежим от константных гиперпараметров (число итераций), которое может быть плохим для одного размера доски и очень плохим для другого.

7 Tabu-search

Будем помечать ситуации, в которые мы уже попадали, чтобы обратно не возвращаться в локальные минимумы, в которых мы не смогли ничего хорошего найти.

Будем ограничивать на следующие L итераций перестановку фигуры i в колонку k (аналогично для второй фигуры в паре), которую мы только что совершили. Так мы относительно дешево помечаем ситуации, в которых мы уже были, чтобы не ходить кругами в одном локальном минимуме.

Казалось бы, данное решение должно было дать сильный прирост качества, т.к. решает основную нашу проблему. Однако это не улучшило результаты, а скорее даже ухудшило.

8 Почему так плохо

Почти ни к одному из рассмотренных подходов не было комментария о том, насколько эффективно он решал задачу, т.к. +- все подходы одинаково плохо решали задачу. Очевидно, что результаты должны были быть лучше и, вероятно, в коде была допущена серьезная ошибка, поэтому я просто описал методы, которые были опробованы. Потенциально лучшее решение должно было быть достигнуто на рестартах локального поиска (мб отжига) + tabu-search без ограничений на итерации и со всеми точками останова (мб стоило бы добавить их еще, но это никакие идеи не растили качество, поэтому я уже не стал пробовать).

Первое, что приходило в голову - неправильно считается число коллизий. Брутфорсом я протестировал на случайных досках и их случайных соседях подсчет коллизий, ошибок не было выявлено. Возможно, я неаккуратно использовал std::random в ++, либо при разработке менеджера доски допустил какую-нибудь специфичную ошибку, связанную с памятью. Может быть проблема была в том, что я делаю много лишних операций в целях инкапсуляции менеджера доски, поэтому глобальное число итераций на порядок меньше, чем можно было бы достичь.

Очень жаль, что не вышло обнаружить причину такого плохого качества и получить хорошие результаты. В письме указана ссылка на лучшую посылку (8 тестов) и ссылка на актуальную версию кода, где есть имплементация оставшихся подходов.