AMAN

Dataset

Есть готовый dataset для этой задачи, на который часто ссылаются в статьях.

Обозначения и сокращения

FCFS (First-Come-First-Served) - порядок посадки, при котором самолеты садятся в том же порядке в котором они прилетают без внесения каких-либо изменений.

N - число самолетов.

M - число посадочных полос.

K - ограничение на максимальный сдвиг самолета в итоговом порядке посадки, относительно его позиции в FCFS.

DP

Очевидно, что в общем виде задача динамикой за адекватное время не решается. Как правило, вводится ограничение на максимальный сдвиг относительно FCFS (First-Come-First-Served). Зачастую указывается, что данное ограничение делает решение только лучше: операторам проще перестраивать самолеты. Но кажется, что это утверждение не совсем справедливо - во-первых, люди часто хотят сделать свою идею более значимой, во-вторых должны быть автоматизированные способы это проделать.

Дальше используем обозначения: K - максимальный сдвиг, N - число самолетов.

Параметры:

1. Маска севших самолетов, с учетом k не все маски допустимы, а общее число масок слишком велико, поэтому этот параметр реализуем

через мапу. Пример недопустимой маски при k = 3: 1100001 - 7-ой самолет сел раньше, чем 4-м.

2. Время освобождения полосы, количество возможных значений зависит от частоты дискретизации по времени.

DP. Функция штрафа

В случае с монотонной функцией штрафа (неубывающей) самолеты нужно сажать в минимально возможное время. Поэтому, из одного состояния существует не более, чем k переходов.

В случае с унимодальной функцией штрафа время посадки перебирается с нужной частотой дискретизации до минимума функции штрафа, после минимума всегда выбирается минимально возможное.

В случае с произвольной функцией штрафа нужно перебирать все возможные времена посадки.

Для монотонной и унимодальной функции штрафа разумно также использовать мапу для хранения параметра.

DP. Классы турбулентности

Если учитывать классы турбулентности честно, то необходимо хранить еще один параметр: класс турбулентности последнего севшего самолета. Можно попробовать учитывать класс турбулентности как требуемое для посадки самолета, непонятно насколько это возможно (во всех источниках, которые я встретил время между посадками задается не временем, а дистанцией). В любом случае возможно выбрать такие времена, требуемые для посадки, что будет выдержана достаточная пауза, но это может снизить точность решения.

DP. Несколько посадочных полос

Вместо одного параметра времени освобождения полосы имеем M таких параметров.

DP. Интересные идеи

1. Bayen AM, Tomlin CJ, Ye Y, Zhang J (2004) An approximation algorithm for scheduling aircraft with holding time. In: 43rd IEEE conference on

decision and control, Atlantis, Paradise Island, Bahamas

Ищут 5-приближение для суммы времен посадки.

2. Chandran B, Balakrishnan H (2007) A dynamic programming algorithm for robust runway scheduling.

Учитывают неточность времени посадки, считают вероятность того, что будет нарушено ограничение на паузу между посадками.

DP. И зачем все это?

Во-первых можем получить достаточно простое решение для сравнения, которое лучше, чем FCFS.

Во-вторых можно применять где-то в методе ветвей и границ: первоначальная верхняя оценка, оценки различных эвристик.

Метод ветвей и границ

Пока привожу реализацию из достаточно старой статьи, уверен, что есть интереснее, но пока хочется посмотреть еще и на другие методы.

Вершины в дереве решений аналогичны состояниям в динамике, но их не нужно хранить явно. Делаем обход в глубину, отсекаем ветви в которых уже накоплен штраф больше, чем текущая оценка.

Для ускорения процесса, делаем нечто похожее на K:

1. Есть параметры: максимальная глубина D, количество фиксируемых самолетов на каждой итерации - S. $S < D$.

2. Запускаем алгоритм на первых D самолетах, фиксируем первые S самолетов из лучшей полученной перестановки.

3. Повторяем пока не рассмотрим все самолеты.

Оценки снизу

Если применять метод ветвей и границ, то хочется иметь еще и оценки снизу - получаем дополнительные отсечения + возможность обходить дерево с учетом приоритетов - как в A*.

Простой и быстрый, но очень неточный способ оценки: возьмем для каждого самолета минимум его функции штрафа и просуммируем. Требует O(N) времени. Можно реализовать за O(log(N)) если учесть, что подмножество самолетов для которых нам нужно посчитать оценку снизу на каждом запросе отличается от предыдущего на один элемент (ДО). Кажется,

что применять его можно только для отсечения ветвей, но никак ни для приоритетов.

Пусть мы рассмотрели некоторую вершину $S1(U, t_{s1,1}, ..., t_{s1,M}, C)$, где U - подмножество еще севших самолетов, $t_{s1,i}$ - время освобождения i-ой посадочной полосы, C - класс турбулентности последнего севшего самолета и знаем для нее цену f(S1) посадки всех самолетов из множества U. Тогда для вершины $S2(U, t_{s2,1}, ..., t_{s2,M}, C)$, такой, что $\forall i : t_{s1,i} \leq t_{s2,i}$ выполняется условие $f(S1) \leq f(S2)$. Пока непонятно, как быстро искать максимум по всем таким состояниям S1, для данного S2. Еще кажется, что можно ослабить требование на полное совпадение U и C.

Local search euristic

Встретил этот алгоритм в одной из статей про генетические алгоритмы, результат у него лучше, чем у генетических алгоритмов, но работает он намного дольше (справляется с тестами на 44 самолета, но не справляется с 50, хотя причины могут быть в несколько специфичных тестах для N = 50 - только в них FCFS совсем не справляется с задачей без нарушений ограничений на паузу).

Идея: есть перестановка самолетов, умеем для перестановки считать штраф, выбираем 2 позиции в перестановке и меняем их местами, если результат стал лучше - фиксируем, повторяем пока есть возможность улучшить.

Генетические алгоритмы

Как я уже писал существует очень много их вариаций для нашей задачи.

Генетические алгоритмы. Хромосомы - перестановки

1. Каждая особь задается одной хромосомой. Оператор скрещивания не используется, порождение новых особей только в результате мутаций. Для мутаций выбираются либо случайные особи, либо случайные с большей вероятностью у лучших особей. Сами мутации, либо shuffle какой-то части перестановки, либо swap двух случайных позиций. Утверждается, что вариант с приоритетным выбором лучших особей сходится намного быстрее. Для нескольких посадочных полос расстановка по полосам производится жадно.

2. Каждая особь задается двумя хромосомами. Вторая хромосома обозначает какая полоса предназначена для каждой позиции в перестановке. Мутации идентичны (вторая хромосома при этом не трогается). Скрещивание производится только по второй хромосоме. Утверждается, что сходится медленнее чем первый способ.

Генетические алгоритмы. Гены - времена посадки

Пока лучшее из того, что встречал.

Для каждого самолета время посадки задается числом $P \in [0, 1]$. Данное число представляет время посадки относительно минимально возможного, отнормированное по длине интервала для посадки.

Хромосома - для каждого самолета храним P и посадочную полосу. Возможно наличие особей нарушающих необходимую паузу между самолетами. Оценка особи - пара (fitness, unfitness). Fitness - наша функция штрафа. Unfitness - равно нулю, если особь выдерживает паузы, иначе больше нуля (детали пока не разбирал). Сравнение - сначала по unfitness, потом по fitness.

Оператор скрещивания:

1. Может быть несколько родителей.

2. Величины P определяются как взвешенная комбинация P родителей. Веса случайны.

3. Полосы посадки выбираются случайно, среди тех, что есть для родителей.

При этом сам алгоритм несовсем типичный: используются эвристики для улучшения особи, выбор родителей не совсем случайный. С этим пока не до конца разобрался.

На всех тестах отработал не хуже, чем Local search euristic, за исключением тех, где Local search euristic работало неадекватное время. Время работы (из худших на тестах ожидаемой размерности) - 20 секунд. Алгоритм запускали 10 раз и выбирали лучший результат.

Алгоритм имитации отжига

Нашел одну статью в которой достаточно успешно применили этот алгоритм.

В качестве точек использовали перестановки. В качестве операторов: либо swap, либо insert (выбираем случайную позицию и вставляем ее на новое место).

Привиденные результаты полностью аналогичны предыдущему алгоритму, но время в 10 раз лучше.