Критериальники по ТПР-2012 by theflybat. v. 0.1

1. Метод динамического программирования на примере распределительной задачи.

   Задача: n предприятий производят $f_n(y)$ продукта, если им скормить y единиц некоторого сырья. Количество сырья ограничено: $\sum_i y_i \le Y_{lim}$. Цель: максимизировать количество конечного продукта и уложиться в отведенное количество ресурса: $\sum_i f_i \to max$.
   (Можно рассказать задачу о студенте и экзаменах).
   Применение ДП: решить подзадачу о распределении $y \le Y_{lim}$ ресурса одному предприятию, потом двум, потом n. (Динамическое программирование – алгоритмический вариант рассуждения по индукции). Потом после нахождения $f_n(Y_{lim})$ при помощи обратного хода найти оптимальные значения количества сырья для каждого предприятия.

2. Алгоритмы для задачи о рюкзаке с гарантированной точностью 50% и 75%.

   50%. $z^{MG} := \max\{z^G\ \left(\text{результат жадного алгоритма: берем предметы с наибольшим отношением } \frac{\text{цена}}{\text{вес}}\right),\ max\{\text{цена предмета в наборе}\}\}$.
   75%. $z^A := max\{\text{цена предмета в наборе}\}$. Перебираем все пары предметов. Для каждой пары: Если они входят в рюкзак – берем, иначе – не берем. Если взяли, то жадным алгоритмом z^{MG} пытаемся запихать остальные предметы в рюкзак. Если новая стоимость рюкзака ($p_i + p_k + z^{MG}$) больше рекорда z^A, меняем рекорд. После перебора всех пар – данный рекорд – решение задачи о рюкзаке с гарантированной точностью 75%.

3. Аппроксимационные схемы. Полиномиальные и полностью полиномиальные аппроксимационные схемы.

   Алгоритм А – $\varepsilon-$ аппроксимационная схема, если для $\forall\varepsilon \in [0;1]$ он является (1-ε)-приближенным алгоритмом.
   $\varepsilon-$ аппроксимационная схема – полиномиальна, если ее трудоемкость полиномиально зависит от длины записи данных задачи (фиксируем ε).
   $\varepsilon-$ аппроксимационная схема – полностью полиномиальна, если ее трудоемкость полиномиально зависит от длины записи данных задачи и величины $\frac{1}{\varepsilon}$.

4. Задача упаковки в контейнеры, отрицательный результат об аппроксимируемости.

   Найти разбиение некоторого множества предметов с весами < 1 на минимальное число подмножеств-контейнеров, максимальный вес контейнера = 1.
   Отрицательный результат: для $\forall\varepsilon > 0$ существование приближенного полиномиального алгоритма А с гарантированной точностью $R_A = \frac{3}{2} - \varepsilon$ влечет P=NP.
   (Определение гарантированной точности для задачи на минимум: $R_A = \inf\{r \ge 1 \mid \exists N > 0: \frac{A(L)}{OPT\{L\}} \le r \text{ для } \forall L: OPT(L) \ge N\}$).

5. Нижние оценки Martello && Toth.

   Введем $0 \le \alpha \le 0.5$. Все веса предметов $0 \le \text{вес} \le 1$.
   Рассортируем предметы: Small (вес $\le \frac{1}{2}$), Medium ($\frac{1}{2} \le$ вес $\le 1-\alpha$), Large (вес $\ge 1-\alpha$).
   Две нижние оценки для OPT(L):

   - Каждый предмет среди больших\средних требует отдельный контейнер, т.е. в решении не менее $|Large| + |Medium|$ контейнеров. Мелкие предметы лежат либо в отдельных контейнерах, либо вместе с предметами среднего размера. В Medium-контейнерах останется места $|Medium| - \sum_{\text{средних предметов}} \text{веса}$. Следовательно, для мелких предметов потребуется как минимум $\left\lceil \sum_{\text{мелких предметов}} \text{веса} - (|Medium| - \sum_{\text{средних предметов}} \text{веса}) \right\rceil$ контейнеров.
   - Вес каждого маленького предмета заменяется на α. В один контейнер их войдет $\left\lfloor \frac{1}{\alpha} \right\rfloor$ штук. Тогда потребовалось бы $\left\lceil |Small| \Big/ \left\lfloor \frac{1}{\alpha} \right\rfloor \right\rceil$ контейнеров.
     Но часть таких предметов можно упаковать в пустое пространство i-го Medium-контейнера, который будет иметь $[1 - w_i \text{ вес } i-\text{го среднего предмета}]$ свободного места, и куда войдет $\left\lfloor \frac{1-w_i}{\alpha} \right\rfloor$ мелких предметов. Распределение контейнеров для больших\средних предметов такое же, как в предыдущем пункте.

6. Метод генерации столбцов для задачи упаковки в контейнеры.

   Выбираем некоторый вариант упаковки в контейнеры $J' \in J$, для которого подзадача ЛП имеет решение (применяем жадные эвристики First-Fit, Next-Fit …).
   Находим для этой подзадачи ЛП и двойственной к ней пару ОРТ – решений: (x_j^*, λ_j^*).
   Решаем для λ_j^* задачу о рюкзаке, вычисляем α. ОРТ решение этой задачи даст новый вариант упаковки контейнера.

   - Если $\alpha \le 1$, то $\overline{x_j} = \begin{cases} x_j^*, j \in J' \\ 0, j \in J \backslash J' \end{cases}$ - ОРТ решение для всего J.
   - Иначе получен новый вариант упаковки, добавляем его в J'

7. Задача календарного планирования. Критические работы, пути, критическое время проекта.

   Задача – сформировать расписание работ с минимальными затратами\временем выполнения\другими свойствами.
   Критическая работа – ее задержка приведет к задержке всего проекта.
   Критическое время проекта $T_{critical}$ – наиболее раннее время завершения всего проекта.
   Критический путь – всякий путь в графе работ\событий, имеющий длину $T_{critical}$.

8. Постановка задачи календарного планирования с ограниченными ресурсами.

   Задача – найти допустимое расписание, укладывающееся в ресурсные и директивные ограничения и с минимальным временем окончания всех работ.

Ресурс – возобновляемый (не использовал – пропал, выдается периодически), невозобновляемый (когда захотел – тогда и использовал, дается только один раз), складируемый (не использовал – можно отложить и использовать накопленный ресурс позже.)

9. Задача коммивояжера. Теорема о погрешности приближенных полиномиальных алгоритмов и алгоритмов локального спуска.

Задан граф вида города\цена маршрута. Задача – обойти все города по одному разу с минимальной ценой поездки (построить гамильтонов цикл минимального веса в данном графе).
Теорема 1: Если существует приближенный полиномиальный алгоритм А, который для любой задачи коммивояжёра дает $A \leq const * OPT$, $1 \leq const \leq \infty$, то P=NP.
Теорема 2: Пусть А – алгоритм локального спуска и окрестность цикла имеет полиномиальную мощность. Тогда если для любой задачи коммивояжёра $A \leq const * OPT$, $1 \leq const \leq \infty$, то P=NP.

10. Задача коммивояжера с неравенством треугольника. Алгоритм с гарантированной оценкой точности 2.

Задача коммивояжера с неравенством треугольника: для матрицы расстояний С: $c_{ij} \leq C_{ik} + c_{kj}, 1 \leq i, j, k \leq n$.
Алгоритм:

- построить остовное дерево (алгоритм Краскала\Прима).
- модификация дерева: обходим дерево в глубину, помечаем вершины, если уже помечена, пропускаем ее, идем дальше пока не вернемся в начальную вершину или не найдем непомеченную. Цепочку дуг для помеченных вершин заменяем прямой дугой в непомеченную или первую вершину.

11. Нижние оценки в задаче коммивояжера: примитивная оценка, оценка линейного программирования, оценка задачи о назначениях, минимальные 1-деревья.

Примитивная: плата за выезд a_i – минимальный вес выходящего из города ребра, плата за въезд b_j – минимум из (стоимость поездки из другого города в этот – стоимость выезда из другого города). $OPT \geq \sum_i a_i + \sum_j b_j$.
Оценка ЛП: заменим в модели задачи x_{ij} (едем из i в j) $\in \{0,1\}$ на непрерывный интервал $0 \leq x_{ij} \leq 1$. Полученная задача ЛП дает оценку не хуже предыдущей.
1-Деревья: ослабим условия для гамильтонова цикла: заменим условие инцидентности каждой вершины ровно двум ребрам на условие инцидентности одной заданной вершины ровно двум ребрам => получим нижнюю оценку. Удаляем заданную вершину, строим минимальное остовное дерево, добавляем 2 ребра минимального веса, проходящих ч\з заданную вершину – получим 1 – дерево.
Задача о назначениях (расставить рабочих по станкам с min суммарным рабочим временем): opt решение данной задачи дает нижнюю оценку для задачи коммивояжера не хуже, чем оценка 1-деревьев.

12. Алгоритм решения задачи о назначениях.

Алгоритм состоит из нескольких одинаковых этапов. На каждом этапе дефект матрицы (число столбцов без минимальных элементов) уменьшается на 1. Пока дефект не 0:

- выбираем столбец без альфа-основы (минимального элемента).
- увеличиваем его так, чтобы все прочие альфа-минимальные элементы остались альфа-минимальными (получим альтернативную основу в строке).
-

(я нифига не понял =\)

13. Метод ветвей и границ для задачи коммивояжера.

Имеется множество допустимых решений задачи D, для каждого его подмножества умеем вычислять нижнюю\верхнюю оценки. Сначала вычисляем Up(D), Bottom(D), если они совпали – нашли opt, выходим. Иначе вычисляем начальный рекорд Rec=Up(D), разбиваем D на подмножества, для которых также вычисляем верхнюю\нижнюю оценки. Если для некоторого подмножества его верхняя оценка > рекорда, тогда мы выкидываем это подмножество, если нижняя оценка < рекорда, меняем рекорд, и продолжаем процесс дробления множеств. Т.к. D – конечное множество, то процесс конечен и в результате дает точное решение задачи.
Для задачи коммивояжера разбиение представляем в виде бинарного дерева, где каждой вершине соответствует список из посещенных и запрещенных городов. Нижнюю оценку можно взять примитивной, верхнюю – алгоритм «иди в ближайший не пройдённый город». Основная идея – угадать opt решение на подмножестве и ветвиться уже от него.

14. Классификация задач теории расписаний. Примеры.

Машины: характеристика машин: каждой работе – своя машина; машины параллельны и одинаковы; машины параллельны, с разными скоростями работы; машины параллельны, длительности работ произвольны; рабочий цех (у каждой операции – своя машина, порядок выполнения – линеен), потоковая линия (каждая работа проходит ч\з упорядоченные машины), открытая линия (на множестве операций нет условия предшествования и может выполняться на нескольких машинах), смешанный цикл (микс рабочего цеха и открытой линии), общий случай (произвольный порядок предшествования операций, как в календарном планировании) + число машин.
Работы: разрешаются\не разрешаются прерывания, работы с условиями предшествования, работы с указанием времени поступления на обслуживание, работы с уточнением для времени выполнения операций, работы с указанием директивных сроков завершения работ, групповые (batch) работы.
Целевые функции: время окончания все работ, запаздывание\отклонение\опережение относительно директивных сроков.

Примеры:

- P|prec, $p_i = 1|C_{max}$ – поиск расписания с min временем окончания работ, на m || машинах, с длительностями работ 1 и условиями предшествования.
- $1|r_i$, pmtn$|L_{max}$ – задача на одной машине, с возможностью прерывания работ, директивными сроками окончания и произвольными временами появления работы.
- J3$|p_{ij} = 1|C_{max}$- задача «рабочий цех», поиск min времени окончания всех работ на 3 машинах, длительности операций = 1, у каждой работы есть свое множество операций, для каждой операции указана машина для ее выполнения.
- R3$|d_i|D_{max}$ – расписание, минимизирующее max отклонение времен завершения работ от директивных сроков на 3 || машинах.
- 1|s-batch$|\sum w_i c_i$ – собрать работы в группы для обработки на 1 машине, так, чтобы min взвешенную сумму окончания всех работ.

15. Алгоритм Лаулера для задачи 1|prec|C_{max}.

Идея: в opt решении (перестановке) последней работой будет работа, не имеющая зависимостей, и перестановка даст $\min_{i \in N} f_i(\sum_{i \in N} p_i)$.
Алгоритм Лаулера: строим перестановку с конца – сначала в хвост ставим элементы, с минимальной ЦФ от времени исполнения и не имеющие последователей, для остальных элементов пересчитываем число последователей и строим перестановку дальше.

16. Алгоритм решения задачи P|ptmn|C_{max}.

Строим нижнюю оценку Low = (max(max время задачи, среднее время выполнения n задач на m машинах)).
Алгоритм: возьмем произвольный список работ, загружаем машины последовательно в интервале [0;Low]. Если последняя работа не влезла, остаток передаем на следующую машину и т.д. Получим расписание, причем оно будет opt.

17. Алгоритм решения задачи Джонсона F2||C_m

Алгоритм построения opt перестановки: разобьём множество работ J на 2 множества $J_1 = \{j \mid a_j \le b_j\}$,$J_2 = \{j \mid a_j > b_j\}$. Далее множество J_1 упорядочим по возрастанию времен выполнения работ, J_2, соответственно, по убыванию. Если разбиение на множестве работ J с такими свойствами найдено, то получена оптимальная перестановка работ.

18. Задача о покрытии, алгоритм Хвatала, оценка его погрешности, экстремальный пример

Задача: покрыть определенный набор участков постами ГАИ (например), с максимальной эффективностью (min число постов для контроля всех опасных участков).
Идея алгоритма: на каждом шаге алгоритма выбирается такой пост, который покрывает максимальное число еще не покрытых участков.
Погрешность: введем $H(p) = \sum_{i=1}^{p} \frac{1}{i}$. Для приближенного APR и точного OPT решений справедливо соотношение: $\sum_{i \in APR} c_i \le H(\max_{i \in I} \sum_{j \in J} a_{ij}) \sum_{i \in OPT} c_i$
Пример: n участков, $n+1$ пост, со стоимостями открытия $[\frac{1}{n}, \dots, \frac{1}{2}, 1, 1+\varepsilon]$, причем каждый пост от 1 до N покрывает только один «свой участок». Если расставить N+1 постов, то при затратах $(1-\varepsilon)$ будут покрыты все участки за минимальную цену. Жадный алгоритм же возьмёт N постов, которые в совокупности получатся дороже одного поста: $(1+\varepsilon) < 1 + \frac{1}{2} + \dots \le 1 + logN$.

19. Задачи размещения в условиях конкуренции, их связь с принятием решений голосованием, безнадежный пример.

Задача: найти схему размещения предприятий, такую, чтобы при наличии конкурента и избирательных клиентов получать максимальную прибыль.
Пример: треугольник с 3 вершинами-потребителями, 6 предприятиями и возможностью открыть только одно предприятие.
Связь: можно построить мат. модель размещения предприятий с учетом предпочтений клиентов (клиент голосует, к кому из предпринимателей он пойдет).

20. Матричные игры. Определение седловой точки. Необходимые и достаточные условия равенства верхней и нижней цен игры в чистых стратегиях. Теорема фон-Неймана. Дилемма заключенных.

Матричная игра: игра с нулевой суммой и конечным числом чистых стратегий. Матрица задает выигрыш\проигрыш.
Седловая точка: пара $(\{i_0, j_0\} | \forall i, j: a_{ij_0} \le a_{i_0 j_0} \le a_{i_0 j})$.
Условия: существование седловой точки.
Теорема: В матричной игре существует пара смешанных стратегий (p^*, q^*), таких, что:

- $E(p, q^*) \le E(p^*, q^*) \le E(p^*, q)$ (аналог седловой точки).
- верхняя оценка $=$ нижняя оценка $= E(p^*, q^*)$.

Дилемма: сдашь – и тебя сдадут, идеальное решение – всем помолчать, но рациональные игроки всегда будут пытаться сдать другого, что будет приводить к нерациональному решению.