

Исследование операций

Лекция. Приближенные алгоритмы 2

Виктор Васильевич Лепин

Линейное программирование как метод вычисления нижней границы *OPT*

Отметим, что: $z_{LP} \leq z_{ILP} = OPT$

- **Шаг 1:** Сформулируйте рассматриваемую задачу как задачу целочисленного линейного программирования (ILP);

- **Шаг 1:** Сформулируйте рассматриваемую задачу как задачу целочисленного линейного программирования (ILP);
- Введем 0/1 переменную x_j для каждого подмножества S_j , где $x_j = 1$ означает выбор S_j , и 0 означает отказ.

- **Шаг 1:** Сформулируйте рассматриваемую задачу как задачу целочисленного линейного программирования (ILP);
- Введем 0/1 переменную x_j для каждого подмножества S_j , где $x_j = 1$ означает выбор S_j , и 0 означает отказ.
- ILP модель:

$$\begin{array}{ll} \min & z = \sum_{j=1}^m w_j x_j \\ \text{s.t.} & \sum_{j:e \in S_j} x_j \geq 1 \quad \forall e \in U \\ & x_j = 0/1 \quad \forall i \end{array}$$

- **Шаг 1:** Сформулируйте рассматриваемую задачу как задачу целочисленного линейного программирования (ILP);
- Введем 0/1 переменную x_j для каждого подмножества S_j , где $x_j = 1$ означает выбор S_j , и 0 означает отказ.
- ILP модель:

$$\begin{array}{ll} \min & z = \sum_{j=1}^m w_j x_j \\ \text{s.t.} & \sum_{j:e \in s_j} x_j \geq 1 \quad \forall e \in U \\ & x_j = 0/1 \quad \forall i \end{array}$$

- Т.о. мы имеем $OPT = z_{ILP}$, где z_{ILP} — оптимальное значение целевой функции в задаче ILP.

- **Шаг 2:** Релаксация ILP в LP;

- Шаг 2: Релаксация ILP в LP;
- LP релаксация:

$$\begin{array}{ll} \min & z = \sum_{j=1}^m w_j x_j \\ s.t. & \sum_{j:e \in s_j} x_j \geq 1 \quad \forall e \in U \\ & x_j \geq 0 \quad \forall i \\ & x_j \leq 1 \quad \forall i \end{array}$$

- **Шаг 2:** Релаксация ILP в LP;
- LP релаксация:

$$\begin{array}{ll} \min & z = \sum_{j=1}^m w_j x_j \\ \text{s.t.} & \sum_{j:e \in s_j} x_j \geq 1 \quad \forall e \in U \\ & x_j \geq 0 \quad \forall i \\ & x_j \leq 1 \quad \forall i \end{array}$$

- Замечание 1: $z_{LP} \leq z_{ILP} = OPT$, где z_{LP} — оптимальное значение целевой функции в соответствующей задаче LP.

- **Шаг 3:** Поскольку модель LP может генерировать дробное решение, необходимо найти умный способ для преобразования оптимального решения задачи LP (за полиномиальное время) в допустимое решение задачи ILP, имеющее значение целевой функции близкое к оптимальному значению целевой функции в задаче LP.

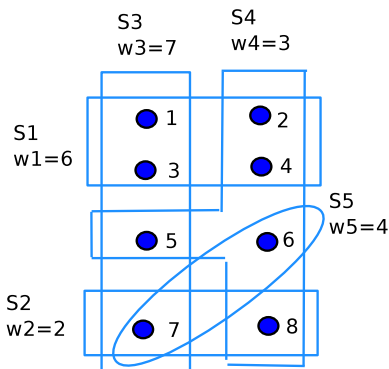
- **Шаг 3:** Поскольку модель LP может генерировать дробное решение, необходимо найти умный способ для преобразования **оптимального решения задачи LP** (за полиномиальное время) в **допустимое решение задачи ILP**, имеющее значение целевой функции близкое к оптимальному значению целевой функции в задаче LP.
- **Сложность:** как получить допустимое решение задачи ILP основываясь на оптимальном решении задачи LP?
Округлять!

Алгоритм LP+Rounding

- 1: Решить задачу LP , чтобы получить ее оптимальное решение x^* ;
- 2: $I = \text{NULL}$;
- 3: **for all** subset S_j **do**
- 4: **if** $x_j^* \geq \frac{1}{d}$ **then**
- 5: $I = I + \{j\}$;
- 6: **end if**
- 7: **end for**
- 8: **return** I ;

- Здесь d обозначает наибольшую степень покрытия элемента из U , т.е. $d = \max_{e \in U} |\{j : e \in S_j\}|$.

Пример



$$\begin{array}{llllllllll}
 \min & 6x_1 & + & 2x_2 & + & 7x_3 & + & 3x_4 & + & 4x_5 \\
 s.t. & x_1 & & & + & x_3 & & & & \geq 1 & \text{(item 1,3)} \\
 & x_1 & & & & & + & x_4 & & \geq 1 & \text{(item 2,4)} \\
 & & & & & x_3 & + & x_4 & & \geq 1 & \text{(item 5)} \\
 & & x_2 & & & & & & + & x_5 & \geq 1 & \text{(item 6)} \\
 & & x_2 & + & x_3 & & & & + & x_5 & \geq 1 & \text{(item 7)} \\
 & & x_2 & & & & + & x_4 & & \geq 1 & \text{(item 8)} \\
 & & & & & & & & & x_i & \geq 0 \\
 & & & & & & & & & x_i & \leq 1
 \end{array}$$

- оптимальное решение задачи LP: $x_1 = 0.5$; $x_2 = 1$;
 $x_3 = 0.5$; $x_4 = 0.5$; $x_5 = 0$;

- оптимальное решение задачи LP: $x_1 = 0.5$; $x_2 = 1$;
 $x_3 = 0.5$; $x_4 = 0.5$; $x_5 = 0$;
- Значение целевой функции LP: 10. ($z_{LP} \leq z_{ILP} = OPT.$)

- оптимальное решение задачи LP: $x_1 = 0.5$; $x_2 = 1$;
 $x_3 = 0.5$; $x_4 = 0.5$; $x_5 = 0$;
- Значение целевой функции LP: 10. ($z_{LP} \leq z_{ILP} = OPT.$)
- Округляем решение: $x'_1 = 1$; $x'_2 = 1$; $x'_3 = 1$; $x'_4 = 1$;
 $x'_5 = 0$;

- оптимальное решение задачи LP: $x_1 = 0.5$; $x_2 = 1$;
 $x_3 = 0.5$; $x_4 = 0.5$; $x_5 = 0$;
- Значение целевой функции LP: 10. ($z_{LP} \leq z_{ILP} = OPT.$)
- Округляем решение: $x'_1 = 1$; $x'_2 = 1$; $x'_3 = 1$; $x'_4 = 1$;
 $x'_5 = 0$;
- Получаем значение целевой функции: $18 \leq d \times z_{LP}$.

Theorem

Алгоритм 1 выдает покрытие множества.

Theorem

Алгоритм 1 выдает покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент e , т.е. $e \notin \bigcup_{j \in I} S_j$.



Theorem

Алгоритм 1 выдает покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент e , т.е. $e \notin \cup_{j \in I} S_j$.
- Тогда для каждого S_j , которое содержит e , мы имеем $x_j^* < \frac{1}{d}$; (Иначе, S_j было бы выбрано Алгоритмом1.)



Theorem

Алгоритм 1 выдает покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент e , т.е. $e \notin \cup_{j \in I} S_j$.
- Тогда для каждого S_j , которое содержит e , мы имеем $x_j^* < \frac{1}{d}$; (Иначе, S_j было бы выбрано Алгоритмом1.)
- Т.о., $\sum_{j: e \in S_j} x_j^* < \frac{1}{d} |\{j : i \in S_j\}| \leq 1$.



Theorem

Алгоритм 1 выдает покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент e , т.е. $e \notin \cup_{j \in I} S_j$.
- Тогда для каждого S_j , которое содержит e , мы имеем $x_j^* < \frac{1}{d}$; (Иначе, S_j было бы выбрано Алгоритмом1.)
- Т.о., $\sum_{j: e \in S_j} x_j^* < \frac{1}{d} |\{j : i \in S_j\}| \leq 1$.
- Получили противоречие ограничениям линейной модели.



Theorem

(Hochbaum '82) Алгоритм1 является d -приближенным для задачи SETCOVER.

Theorem

(Hochbaum '82) Алгоритм1 является d -приближенным для задачи SETCOVER.

Доказательство.

- Алгоритм1 возвращает набор I , имеющий стоимость: $C = \sum_{j \in I} w_j$;

$$\begin{aligned} C &= \sum_{j \in I} w_j \\ &\leq \sum_{j \in I} w_j x_j^* d \\ &\leq \sum_{j=1}^m w_j x_j^* d \\ &= d \times \sum_{j=1}^m w_j x_j^* \\ &= d \times z_{LP} \\ &\leq d \times OPT \end{aligned}$$

Theorem

(Hochbaum '82) Алгоритм1 является d -приближенным для задачи SETCOVER.

Доказательство.

- Алгоритм1 возвращает набор I , имеющий стоимость: $C = \sum_{j \in I} w_j$;

$$\begin{aligned} C &= \sum_{j \in I} w_j \\ &\leq \sum_{j \in I} w_j x_j^* d \\ &\leq \sum_{j=1}^m w_j x_j^* d \\ &= d \times \sum_{j=1}^m w_j x_j^* \\ &= d \times z_{LP} \\ &\leq d \times OPT \end{aligned}$$

- Здесь, первое неравенство следует из “округления”, т.е. $x_j^* \geq \frac{1}{d}$.

Нижняя граница 2: $z_{Dual} \leq z_{LP} \leq z_{ILP} = OPT$

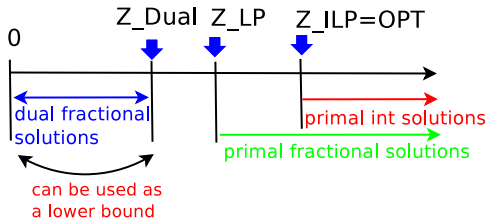
- Идея: округлить двойственное решение.

- Идея: округлить двойственное решение.
- Двойственная задача:

$$\begin{aligned} \max d = & \sum_{e \in U} y_e \\ \text{s.t.} \quad & \sum_{e: e \in S_j} y_e \leq w_j \quad \forall S_j \\ & y_e \geq 0 \quad \forall e \in U \end{aligned}$$

Двойственная задача дает другую нижнюю оценку

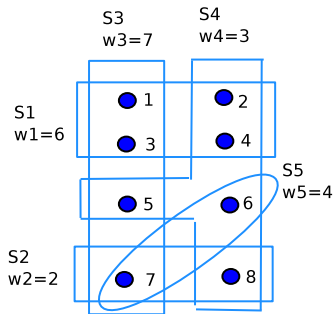
Замечание 2: $\sum_e y_e \leq z_{LP} \leq OPT$ для любого допустимого решения y двойственной задачи;



Dual LP + Rounding Algorithm

- 1: Решить двойственную задачу LP . Пусть y^* ее решение;
- 2: $I = \text{NULL}$;
- 3: **for all** подмножества S_j **do**
- 4: **if** $\sum_{e:e \in S_j} y_e^* = w_j$ **then**
- 5: $I = I + \{j\}$;
- 6: **end if**
- 7: **end for**
- 8: **return** I ;

Пример



$$\begin{array}{llllllllllllll}
 \max & y_1 & + & y_2 & + & y_3 & + & y_4 & + & y_5 & + & y_6 & + & y_7 & + & y_8 \\
 s.t. & y_1 & + & y_2 & + & y_3 & + & y_4 & & & & & & & & \\
 & & & y_2 & & & & & & & & & & y_7 & + & y_8 \\
 & y_1 & & & + & y_3 & & & + & y_5 & & & & + & y_7 & \\
 & & & y_2 & & & + & y_4 & + & y_5 & & & & & + & y_8 \\
 & & & & & & & & & & & y_6 & + & y_7 & & \\
 & & & & & & & & & & & & & & & y_i \\
 & & & & & & & & & & & & & & & \geq 0
 \end{array}$$

- Оптимальное решение Двойственной задачи LP: $y_1 = 6$;
 $y_2 = 0$; $y_3 = 0$; $y_4 = 0$; $y_5 = 0$; $y_6 = 3$; $y_7 = 1$; $y_8 = 0$;

- Оптимальное решение Двойственной задачи LP: $y_1 = 6$; $y_2 = 0$; $y_3 = 0$; $y_4 = 0$; $y_5 = 0$; $y_6 = 3$; $y_7 = 1$; $y_8 = 0$;
- Оптимальное значение целевой функции: 10
($z_{DualLP} \leq z_{LP} \leq z_{ILP} = OPT.$)

- Оптимальное решение Двойственной задачи LP: $y_1 = 6$; $y_2 = 0$; $y_3 = 0$; $y_4 = 0$; $y_5 = 0$; $y_6 = 3$; $y_7 = 1$; $y_8 = 0$;
- Оптимальное значение целевой функции: 10
($z_{DualLP} \leq z_{LP} \leq z_{ILP} = OPT.$)
- Жесткие ограничения: $I = \{1, 3, 4, 5\}$;

- Оптимальное решение Двойственной задачи LP: $y_1 = 6$; $y_2 = 0$; $y_3 = 0$; $y_4 = 0$; $y_5 = 0$; $y_6 = 3$; $y_7 = 1$; $y_8 = 0$;
- Оптимальное значение целевой функции: 10
($z_{DualLP} \leq z_{LP} \leq z_{ILP} = OPT.$)
- Жесткие ограничения: $I = \{1, 3, 4, 5\}$;
- значение целевой функции: $20 \leq d * z_{DualLP}$.

Theorem

Алгоритм2 строит покрытие множества.

Theorem

Алгоритм2 строит покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент \hat{e} подмножествами выбранными в I ;



Theorem

Алгоритм2 строит покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент \hat{e} подмножествами выбранными в I ;
- Тогда для всех S_j , содержащих \hat{e} , мы имеем $\sum_{e:\hat{e}\in S_j} y_e^* < w_j$; (Иначе, S_j должно быть выбрано и элемент e будет покрыт.)



Theorem

Алгоритм2 строит покрытие множества.

Доказательство.

(от противного)

- Предположим, что существует не покрытый элемент \hat{e} подмножествами выбранными в I ;
- Тогда для всех S_j , содержащих \hat{e} , мы имеем $\sum_{e:\hat{e} \in S_j} y_e^* < w_j$; (Иначе, S_j должно быть выбрано и элемент e будет покрыт.)
- Т.о., увеличив $y_{\hat{e}}^*$ (на малое положительное число) мы увеличим значение целевой функции двойственной задачи LP, не нарушая ограничений. Получили противоречие с оптимальностью y^* .



Theorem

Алгоритм2 является d -приближенным.

Theorem

Алгоритм2 является d -приближенным.

Доказательство.

Пусть C — суммарный вес подмножеств, выданных Алгоритмом2 в качестве решения, т.е. $C = \sum_{j \in I} w_j$. Мы имеем:

$$\begin{aligned} C &= \sum_{j \in I} w_j \\ &= \sum_{j \in I} \sum_{e: e \in S_j} y_e^* \\ &\leq d \sum_{e: e \in U} y_e^* \quad (\text{поскольку любое } e \text{ было включено самое большое } d \text{ раз}) \\ &= dz_{Dual \text{ LP}} \\ &\leq dOPT \quad (\text{since } z_{Dual \text{ LP}} \leq z_{LP} \leq z_{ILP}) \end{aligned}$$



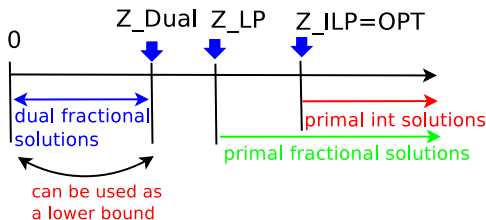
Нижняя граница 3:

величина любого допустимого решения двойственной задачи

$$\leq z_{Dual} \leq z_{LP} \leq z_{ILP} = OPT$$

Алгоритм3: Primal_dual + Rounding

- Идея: допустимое решение двойственной задачи дает нижнюю границу OPT . Т.о., мы можем применить primal_dual стратегию для построения допустимого решения вместо решения двойственной задачи LP.

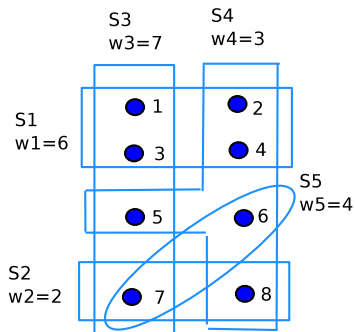


Алгоритм3: Primal_dual + Rounding

Primal_Dual + Rounding

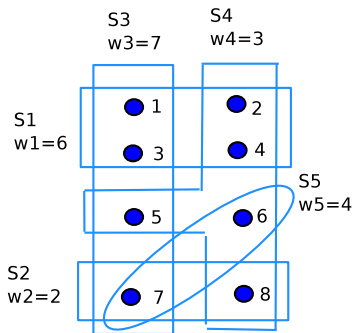
```
1:  $I = \text{NULL};$ 
2:  $y_e = 0$  для всех  $e \in U$ ;
3: while существует не покрытый элемент  $\hat{e}$  do
4:   for all подмножеств  $S_j$  содержащих  $\hat{e}$  do
5:      $g_j = w_j - \sum_{e:e \in S_j} y_e$ ; // вычисляем разрыв
        ограничения  $j$ ;
6:   end for
7:    $i = \arg \min_j g_j$ ;
8:    $y_{\hat{e}} = y_{\hat{e}} + g_i$ ;
9:    $I = I \cup \{i\}$ ;
10: end while
11: return  $I$ ;
```

Пример: Шаг 1



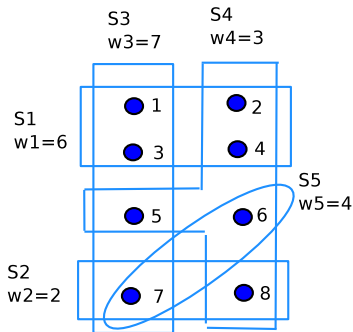
- В начале, мы имеем $y_i = 0$ для всех $1 \leq i \leq 8$.

Пример: Шаг 1



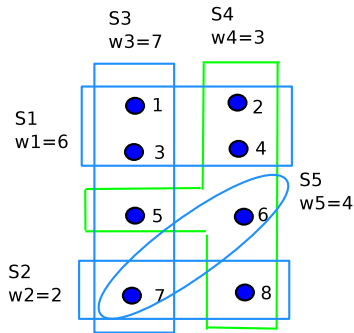
- В начале, мы имеем $y_i = 0$ для всех $1 \leq i \leq 8$.
- Рассмотрим элемент y_2 . Существует два ограничения S_1 и S_4 .

Пример: Шаг 1



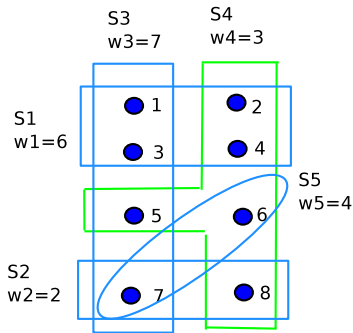
- В начале, мы имеем $y_i = 0$ для всех $1 \leq i \leq 8$.
- Рассмотрим элемент y_2 . Существует два ограничения S_1 и S_4 .
- Увеличим y_2 до 3 (выбирая S_4).

Пример: Шаг 2



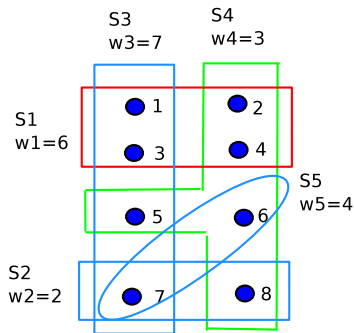
- Рассмотрим элемент y_1 . Имеются два ограничения S_1 and S_3 .

Пример: Шаг 2



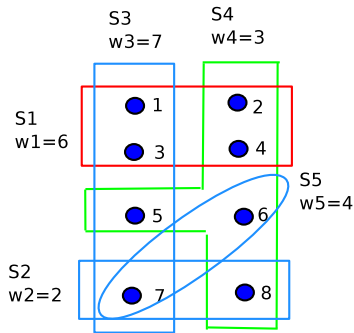
- Рассмотрим элемент y_1 . Имеются два ограничения S_1 and S_3 .
- Увеличим y_1 до 3 (выбирая S_1).

Пример: Шаг 3



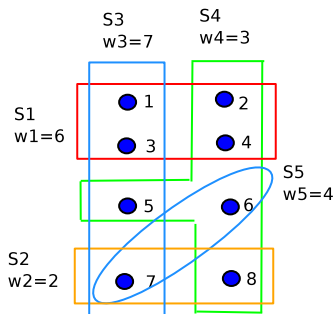
- Рассмотрим элемент y_7 . Есть три ограничения S_2 , S_3 и S_5 .

Пример: Шаг 3



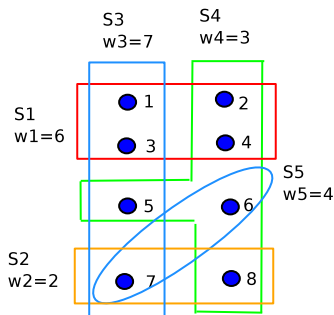
- Рассмотрим элемент y_7 . Есть три ограничения S_2 , S_3 и S_5 .
- Увеличим y_7 до 2 (выбирая S_2).

Пример: Шаг 4



- Все элементы покрыты. Решение есть!

Пример: Шаг 4

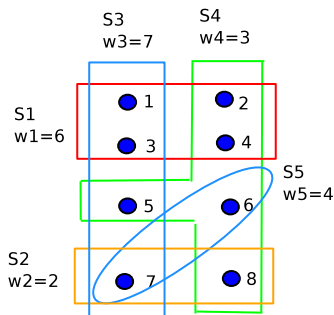


- Все элементы покрыты. Решение есть!

- Двойственное решение:

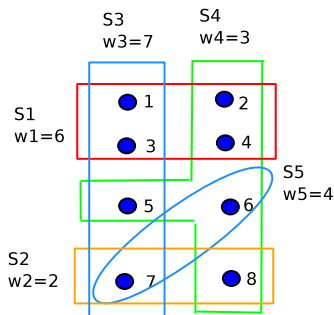
$$y_1 = 3; y_2 = 3; y_7 = 2; y_3 = y_4 = y_5 = y_6 = y_8 = 0;$$

Пример: Шаг 4



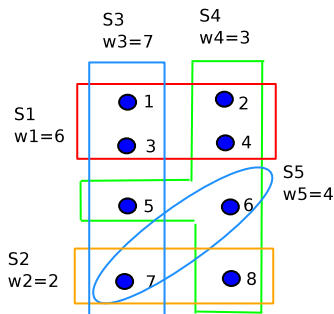
- Все элементы покрыты. Решение есть!
- Двойственное решение:
 $y_1 = 3; y_2 = 3; y_7 = 2; y_3 = y_4 = y_5 = y_6 = y_8 = 0;$
- Значение целевой функции: 8. (Двойственное допустимое решение, $z \leq z_{Dual} \leq z_{LP} \leq z_{ILP} = OPT$)

Пример: Шаг 4



- Все элементы покрыты. Решение есть!
- Двойственное решение:
 $y_1 = 3; y_2 = 3; y_7 = 2; y_3 = y_4 = y_5 = y_6 = y_8 = 0;$
- Значение целевой функции: 8. (Двойственное допустимое решение, $z \leq z_{Dual} \leq z_{LP} \leq z_{ILP} = OPT$)
- Точные ограничения: $I = \{S_4, S_1, S_2\}$

Пример: Шаг 4



- Все элементы покрыты. Решение есть!
- Двойственное решение:
 $y_1 = 3; y_2 = 3; y_7 = 2; y_3 = y_4 = y_5 = y_6 = y_8 = 0;$
- Значение целевой функции: 8. (Двойственное допустимое решение, $z \leq z_{Dual} \leq z_{LP} \leq z_{ILP} = OPT$)
- Точные ограничения: $I = \{S_4, S_1, S_2\}$
- Значение целевой функции:

$$11 \leq d \times z \leq d \times z_{Dual} \leq d \times OPT$$

Lemma

Алгоритм 3 строит допустимое решение y двойственной задачи, и $\sum_{e:e \in S_j} y_e = w_j$ для любого $j \in I$.

Lemma

Алгоритм 3 строит допустимое решение y двойственной задачи, и $\sum_{e:e \in S_j} y_e = w_j$ для любого $j \in I$.

Доказательство.

- База: $y = 0$ — допустимое решение ДЗ, поскольку $\sum_{e:e \in S_j} y_e = 0 \leq w_j$;

Lemma

Алгоритм 3 строит допустимое решение y двойственной задачи, и $\sum_{e:e \in S_j} y_e = w_j$ для любого $j \in I$.

Доказательство.

- База: $y = 0$ — допустимое решение ДЗ, поскольку $\sum_{e:e \in S_j} y_e = 0 \leq w_j$;
- Индукция: предположим, что y является допустимым решением ДЗ до итерации цикла while, т.е. $\sum_{e:e \in S_j} y_e \leq w_j$. Предположим мы увеличили $y_{\hat{e}}$ на g_i , чтобы сгенерировать новое решение y' . Покажем, что y' так же является допустимым решением ДЗ.

$$\begin{aligned}\sum_{e:e \in S_j} \hat{y}'_e &= \sum_{e:e \in S_j} y_e + g_i \quad (g_i \text{ является минимальным}) \\ &= \sum_{e:e \in S_j} y_e + (w_i - \sum_{e:e \in S_i} y_e) \\ &\leq \sum_{e:e \in S_j} y_e + (w_j - \sum_{e:e \in S_j} y_e) \\ &= w_j\end{aligned}$$

Theorem

(Bar-Yehuda, Even '81) Алгоритм 3 является d -приближенным.

Theorem

(Bar-Yehuda, Even '81) Алгоритм3 является d -приближенным.

Доказательство.

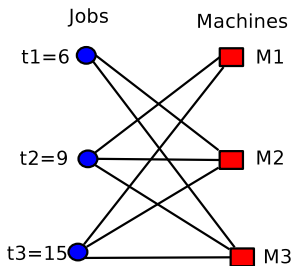
Пусть C — сумма весов подмножеств для решения полученного Алгоритмом3, т.е. $C = \sum_{j \in I} w_j$. Мы имеем:

$$\begin{aligned} C &= \sum_{j \in I} w_j \\ &= \sum_{j \in I} \sum_{e: e \in S_j} y_e \quad (\text{см. строки 9-10}) \\ &\leq d \sum_{e: e \in U} y_e \quad (e \text{ покрывается не более } d \text{ раз}) \\ &\leq dz_{Dual} \quad (y \text{ является ДРДЗ}) \\ &\leq dOPT \quad (\text{поскольку } y \text{ — ДРДЗ}) \end{aligned}$$

Округление. Задача построения расписания для не идентичных параллельных машин

Задача построения расписания для не идентичных параллельных машин

Практическая задача: у нас есть несколько серверов для обработки набора заданий. Однако некоторые сервера не могут выполнять некоторые задания. Как распределить задания по серверам, так чтобы оно было «сбалансированными»?



INPUT:

m машин M_1, M_2, \dots, M_m , n работ $J = \{1, \dots, n\}$ (каждая работа j имеет время выполнения t_j). Для каждой работы j задано подмножество машин $S_j \subset \{M_1, \dots, M_m\}$, которые могут ее выполнить.

OUTPUT:

Назначение работ на машины, которое минимизирует длину расписания, т.е. время завершения выполнения работ, $T = \max_i \sum_{j \in A(i)} t_j$, где $A(i)$ — подмножество работ назначенных для выполнения машиной i ;

(ILP)

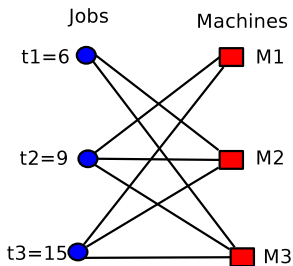
$$\begin{array}{llll} \min & L & & \\ s.t. & \sum_{i=1}^m x_{ji} = t_j & \text{для всех } j \in J & \\ & \sum_{j=1}^n x_{ji} \leq L & \text{для всех } i & \\ & x_{ji} = 0/t_j & \text{для всех } j \in J, i \in S_j & \\ & x_{ji} = 0 & \text{для всех } j \in J, i \notin S_j & \end{array}$$

(Переменная x_{ji} указывает какая часть от t_j назначается для выполнения на машине M_i ;)

(LP)

$$\begin{array}{ll}
 \min & L \\
 \text{s.t.} & \sum_{i=1}^m x_{ji} = t_j \quad \text{для всех } j \in J \\
 & \sum_{j=1}^n x_{ji} \leq L \quad \text{для всех } i \\
 & x_{ji} \leq t_j \quad \text{для всех } j \in J, i \in S_j \\
 & x_{ji} \geq 0 \quad \text{для всех } j \in J, i \in S_j \\
 & x_{ji} = 0 \quad \text{для всех } j \in J, i \notin S_j
 \end{array}$$

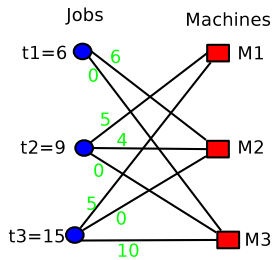
- **Замечание 1:** $z_{LP} \leq z_{ILP} = OPT$
- **Замечание 2:** $\max_j t_j \leq OPT$



(LP)

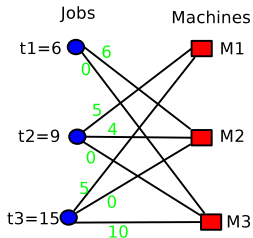
$$\begin{aligned}
 \min z = & \quad L \\
 \text{s.t.} \quad & x_{12} + x_{13} = 6 \\
 & x_{21} + x_{22} + x_{23} = 9 \\
 & x_{31} + x_{32} + x_{33} = 15 \\
 & x_{21} + x_{31} \leq L \\
 & x_{12} + x_{22} + x_{32} \leq L \\
 & x_{13} + x_{23} + x_{33} \leq L \\
 & x_{ji} \geq 0
 \end{aligned}$$

Как построить допустимое расписание из решения релаксационной задачи?



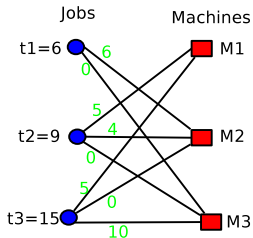
- Предположим, что мы получили решение задачи LP. Как построить допустимое расписание из этого решения?

Как построить допустимое расписание из решения релаксационной задачи?



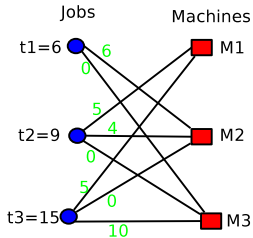
- Напомним, что для любой работы j , мы имеем $t_j = x_{j1} + x_{j2} + \dots + x_{jm}$. Имеется два возможных случая:

Как построить допустимое расписание из решения релаксационной задачи?



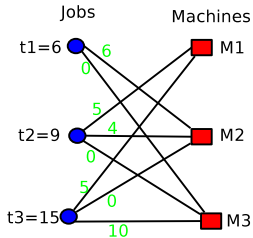
- Напомним, что для любой работы j , мы имеем $t_j = x_{j1} + x_{j2} + \dots + x_{jm}$. Имеется два возможных случая:
 - 1 $\exists i, x_{ji} = t_j$ (называется “целостной работой”): Другими словами, работа j была назначена на машину M_i для полного выполнения. (В примере работа 1).


Как построить допустимое расписание из решения релаксационной задачи?



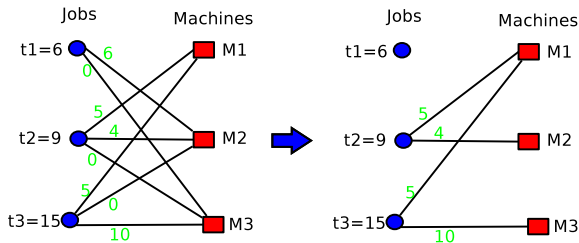
- Напомним, что для любой работы j , мы имеем $t_j = x_{j1} + x_{j2} + \dots + x_{jm}$. Имеется два возможных случая:
 - 1 $\exists i, x_{ji} = t_j$ (называется “целостной работой”): Другими словами, работа j была назначена на машину M_i для полного выполнения. (В примере работа 1).
 - 2 $\forall i, x_{ji} < t_j$ (называется “дробной работой”): Другими словами, работа j была разбита на части, которые назначены различным машинам. В примере две части работы 2 распределены на M_1 и M_2 . □ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶

Как построить допустимое расписание из решения релаксационной задачи?



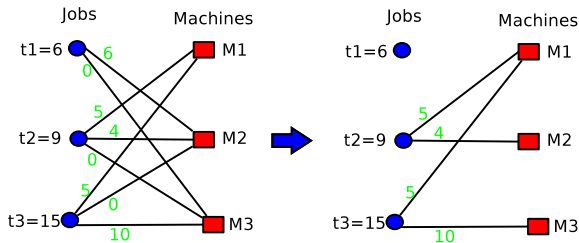
- Напомним, что для любой работы j , мы имеем $t_j = x_{j1} + x_{j2} + \dots + x_{jm}$. Имеется два возможных случая:
 - 1 $\exists i, x_{ji} = t_j$ (называется “целостной работой”): Другими словами, работа j была назначена на машину M_i для полного выполнения. (В примере работа 1).
 - 2 $\forall i, x_{ji} < t_j$ (называется “дробной работой”): Другими словами, работа j была разбита на части, которые назначены различным машинам. В примере две части работы 2 распределены на M_1 и M_2 . 

Две возможности для дробных работ



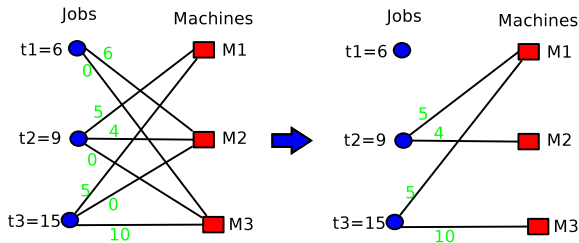
● Действия:

Две возможности для дробных работ



- Действия:
 - Удалить "целостные работы";

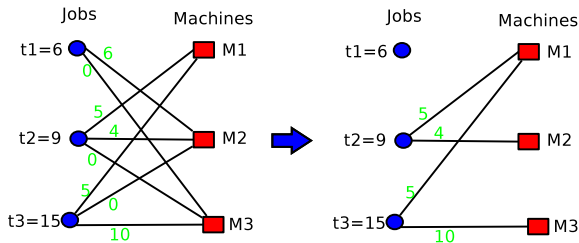
Две возможности для дробных работ



- Действия:

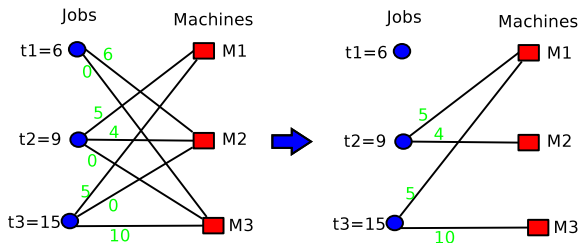
- Удалить "целостные работы";
- Удалить не задействованные назначения, т.е. $x_{ji} = 0$;

Две возможности для дробных работ



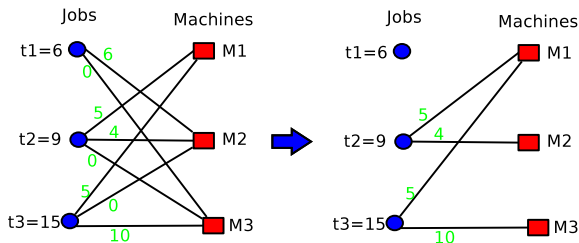
- Действия:
 - Удалить "целостные работы";
 - Удалить не задействованные назначения, т.е. $x_{ji} = 0$;
- Имеются две возможности:

Две возможности для дробных работ



- Действия:
 - Удалить "целостные работы";
 - Удалить **не задействованные назначения**, т.е. $x_{ji} = 0$;
- Имеются две возможности:
 - ❶ Назначение не содержит циклов;

Две возможности для дробных работ



- Действия:
 - Удалить "целостные работы";
 - Удалить **не задействованные назначения**, т.е. $x_{ji} = 0$;
- Имеются две возможности:
 - 1 Назначение **не содержит циклов**;
 - 2 Назначение **содержит один или более циклов**;

Случай 1: Назначение не содержит циклов

- В этом случае, допустимое расписание может быть построено по следующим правилам:

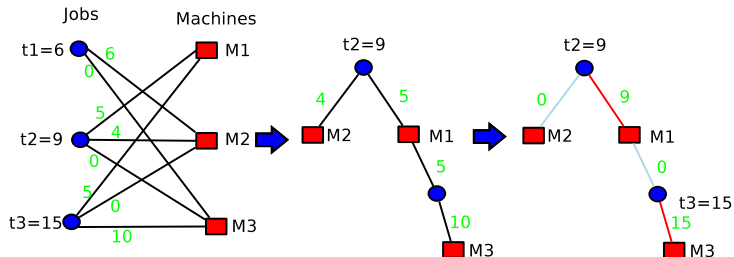
Случай 1: Назначение не содержит циклов

- В этом случае, допустимое расписание может быть построено по следующим правилам:
 - ❶ Округление: Если часть дробной работы j была запланирована для обработки на M_i , мы можем просто запланировать всю работу j на машину M_i .

Случай 1: Назначение не содержит циклов

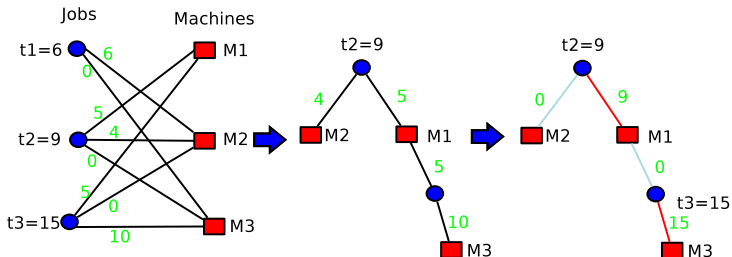
- В этом случае, допустимое расписание может быть построено по следующим правилам:
 - 1 Округление: Если часть дробной работы j была запланирована для обработки на M_i , мы можем просто запланировать всю работу j на машину M_i .
 - 2 Ограничение: каждая машина получает не более одного дробного задания.

Реализация округления



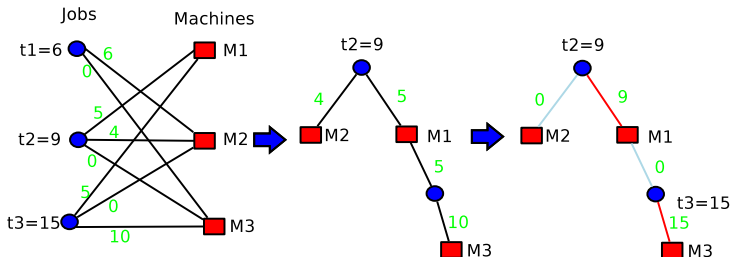
- Операция округления допустима, поскольку:

Реализация округления



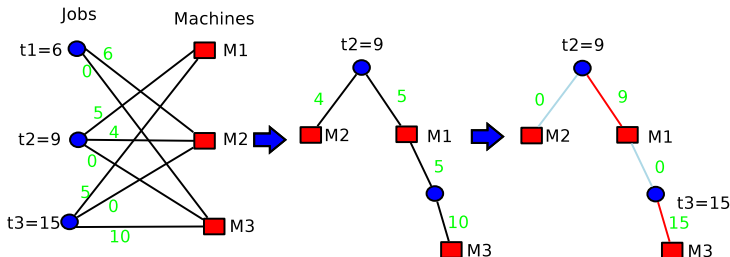
- Операция округления допустима, поскольку:
 - “Отсутствие цикла”. Имеем корневое дерево.

Реализация округления



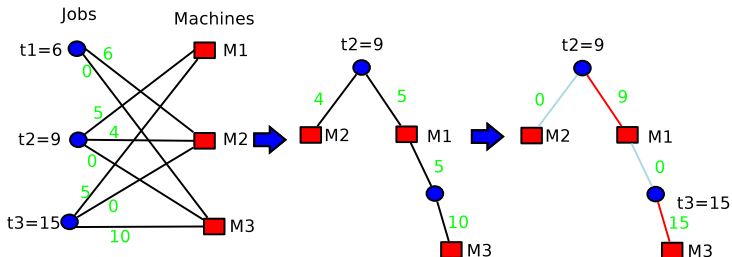
- Операция округления допустима, поскольку:
 - “Отсутствие цикла”. Имеем корневое дерево.
 - Обратите внимание, что любой лист должен быть “машиной”, а не “работой” (Причина: у листа только один родитель. Следовательно, если бы листом являлась бы работа, то она целиком выполнялась родителем; таким образом, это должна быть “целостная работа”)

Реализация округления



- Операция округления допустима, поскольку:
 - “Отсутствие цикла”. Имеем корневое дерево.
 - Обратите внимание, что любой лист должен быть “машиной”, а не “работой” (Причина: у листа только один родитель. Следовательно, если бы листом являлась бы работа, то она целиком выполнялась родителем; таким образом, это должна быть “целостная работа”)
 - Т.о. мы можем просто назначить дробную работу **полностью произвольному** сыну. В примере, назначить J_2 на M_1 , и назначить J_3 на M_2 .

Реализация округления



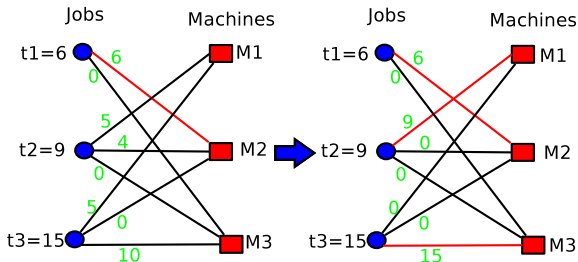
- Операция округления допустима, поскольку:
 - 1 “Отсутствие цикла”. Имеем корневое дерево.
 - 2 Обратите внимание, что любой лист должен быть “машиной”, а не “работой” (Причина: у листа только один родитель. Следовательно, если бы листом являлась бы работа, то она целиком выполнялась родителем; таким образом, это должна быть “целостная работа”)
 - 3 Т.о. мы можем просто назначить дробную работу **полностью произвольному** сыну. В примере, назначить J_2 на M_1 , и назначить J_3 на M_2 .
- Трудоемкость: $O(mn)$.

Случай 1: назначение не содержит циклов

- Скомбинируем "целостные работы получим следующее расписание:

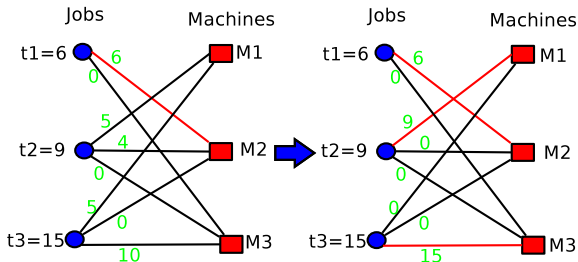
Случай 1: назначение не содержит циклов

- Скомбинируем "целостные работы получим следующее расписание:
- $J_{M1} = \{2\}; J_{M2} = \{1\}; J_{M3} = \{3\}$.



Случай 1: назначение не содержит циклов

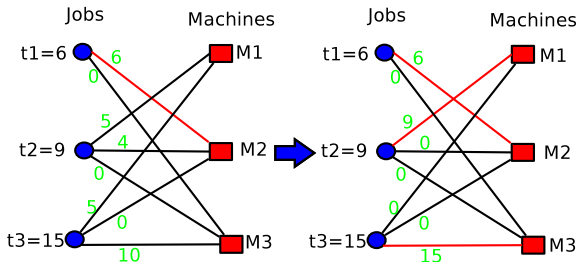
- Скомбинируем "целостные работы получим следующее расписание:
- $J_{M1} = \{2\}; J_{M2} = \{1\}; J_{M3} = \{3\}$.



- $\text{MakeSpan} = 15 \leq 2 \times z_{LP} = 20$

Случай 1: назначение не содержит циклов

- Скомбинируем "целостные работы" получим следующее расписание:
- $J_{M1} = \{2\}; J_{M2} = \{1\}; J_{M3} = \{3\}$.



- $\text{MakeSpan} = 15 \leq 2 \times z_{LP} = 20$
- Можно показать, что стратегия округления не принесет слишком много нагрузки на любую машину.

Theorem

Алгоритм MakeSpanApprox является 2-приближенным.

Theorem

Алгоритм *MakeSpanApprox* является 2-приближенным.

Доказательство.

- Пусть T — длина расписания, построенного алгоритмом *MakeSpanApprox*, и T получено на машине M_i .

$$T = \sum_{j \in J_i} t_j \quad (1)$$

$$= \sum_{j \in J_i, j \neq f_i} t_j + t_{f_i} \quad (2)$$

$$= \sum_{j \in J_i, j \neq f_i} x_{ij} + t_{f_i} \quad (\text{по определению целостной работы}) \quad (3)$$

$$\leq \sum_{j \in J_i} x_{ij} + t_{f_i} \quad (4)$$

$$\leq z_{LP} + t_{f_i} \quad (5)$$

$$\leq z_{LP} + OPT \quad (\text{по замечанию 2}) \quad (6)$$

$$\leq 2OPT \quad (\text{по замечанию 1}) \quad (7)$$

Theorem

Алгоритм *MakeSpanApprox* является 2-приближенным.

Доказательство.

- Пусть T — длина расписания, построенного алгоритмом *MakeSpanApprox*, и T получено на машине M_i .
- Пусть J_i — подмножество работ, назначенных на M_i . J_i состоит из двух частей: целостные работы, назначенные на машину i ($x_{ij} = t_j$), и самое большое **одна** дробная работа (обозначим ее f_i), ($x_{ij} < t_j$). Мы имеем:

$$T = \sum_{j \in J_i} t_j \quad (1)$$

$$= \sum_{j \in J_i, j \neq f_i} t_j + t_{f_i} \quad (2)$$

$$= \sum_{j \in J_i, j \neq f_i} x_{ij} + t_{f_i} \quad (\text{по определению целостной работы}) \quad (3)$$

$$\leq \sum_{j \in J_i} x_{ij} + t_{f_i} \quad (4)$$

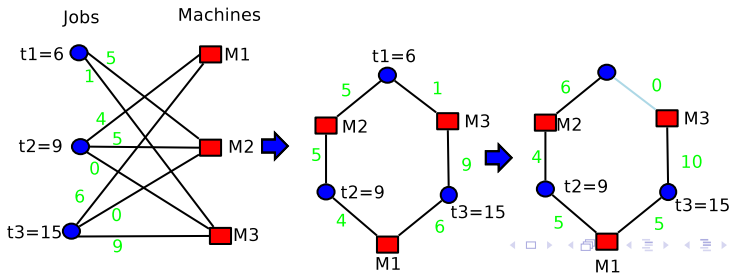
$$\leq z_{LP} + t_{f_i} \quad (5)$$

$$\leq z_{LP} + OPT \quad (\text{по замечанию 2}) \quad (6)$$

$$\leq 2OPT \quad (\text{по замечанию 1}) \quad (7)$$

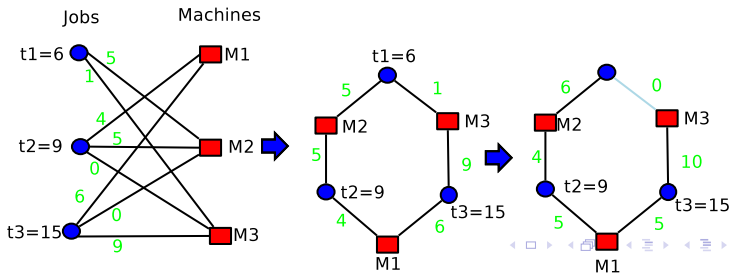
Случай 2: назначение содержит циклы

- Фактически, любой цикл может быть исключен без изменений нагрузки для любой машины, используя следующую “операцию приращения” :



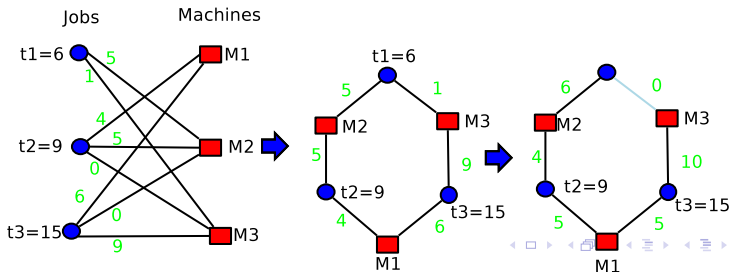
Случай 2: назначение содержит циклы

- Фактически, любой цикл может быть исключен без изменений нагрузки для любой машины, используя следующую **“операцию приращения”** :
 - Рассмотрим цикл C . Найдём наименьшую загрузку (обозначим ее δ);



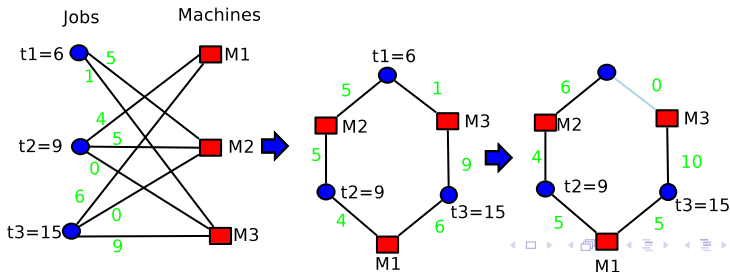
Случай 2: назначение содержит циклы

- Фактически, любой цикл может быть исключен без изменений нагрузки для любой машины, используя следующую **“операцию приращения”** :
 - Рассмотрим цикл C . Найдем наименьшую загрузку (обозначим ее δ);
 - Увеличение / уменьшение нагрузки на δ на ребрах цикла;



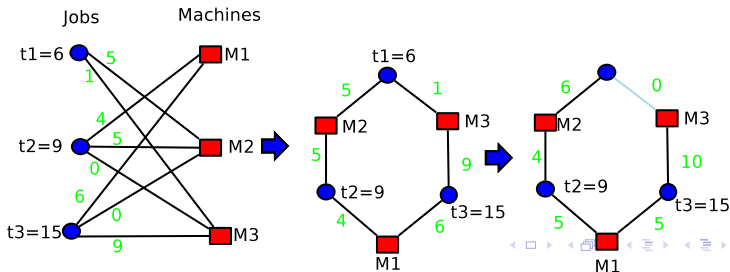
Случай 2: назначение содержит циклы

- Фактически, любой цикл может быть исключен без изменений нагрузки для любой машины, используя следующую **“операцию приращения”** :
 - Рассмотрим цикл C . Найдем наименьшую загрузку (обозначим ее δ);
 - Увеличение / уменьшение нагрузки на δ на ребрах цикла;
 - Таким образом, цикл будет исключен без влияния на нагрузки. (Ребро $J_1 - M_3$ исчезнет и разорвет цикл).



Случай 2: назначение содержит циклы

- Фактически, любой цикл может быть исключен без изменений нагрузки для любой машины, используя следующую **“операцию приращения”** :
 - Рассмотрим цикл C . Найдем наименьшую загрузку (обозначим ее δ);
 - Увеличение / уменьшение нагрузки на δ на ребрах цикла;
 - Таким образом, цикл будет исключен без влияния на нагрузки. (Ребро $J_1 - M_3$ исчезнет и разорвет цикл).
- Трудоемкость: $O(|C|)$ для цикла C . Операция может быть повторена $O(mn)$ раз, чтобы удалить все циклы.



Алгоритм MAKESPANAPPROX

- 1: Решить задачу LP, чтобы получить решение x_{ij} ;
- 2: Удалить циклы в графе с помощью **операции приращения**;
- 3: **for** $i = 1$ to m **do**
- 4: назначить целочтную работу j на машину i , если $x_{ij} = t_j$;
- 5: назначить самое большее **одну** дробную работу машине i ;
- 6: **end for**

Другие полезные техники: масштабирование и параметрическое сокращение

- Масштабирование: округление действительного числа до целого числа; сетка;

- Масштабирование: округление действительного числа до целого числа; сетка;
- Параметрическое сокращение: предположим, что мы уже знаем *OPT*, мы могли бы удалить ненужные части входных данных и тем самым упростить поиск хорошего решения.

Техника масштабирования: от псевдополиномиального алгоритма до PTAS

Дан набор элементов, каждый элемент имеет вес и значение, требуется определить такой набор элементов, чтобы общий вес был меньше заданного предела, а общее значение было как можно большим.

Дан набор элементов, каждый элемент имеет вес и значение, требуется определить такой набор элементов, чтобы общий вес был меньше заданного предела, а общее значение было как можно большим.

Формализованное определение:

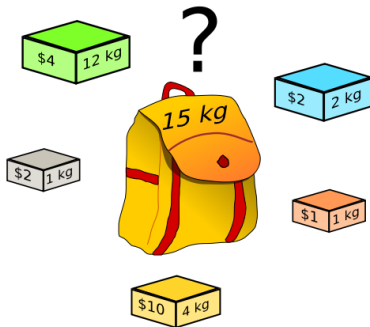
- **Input:**

множество элементов (элемент i имеет вес w_i и значение v_i) и предел общего веса W ;

- **Output:**






множество элементов, которые максимизируют общую стоимость и имеют суммарный вес меньше W

Какое решение наилучшее?







Жадное решение:



(A)



	1.11 \$/kg
	0.58 \$/kg
	0.5 \$/kg
	0.43 \$/kg
	0.4 \$/kg



(B) Max : 15 kg

(1)  → 

(2)  → 

(3)  → 

(4)  → 

(5)  → 

Замечание: Существует два типа алгоритмов динамического программирования для решения Задачи о ранце

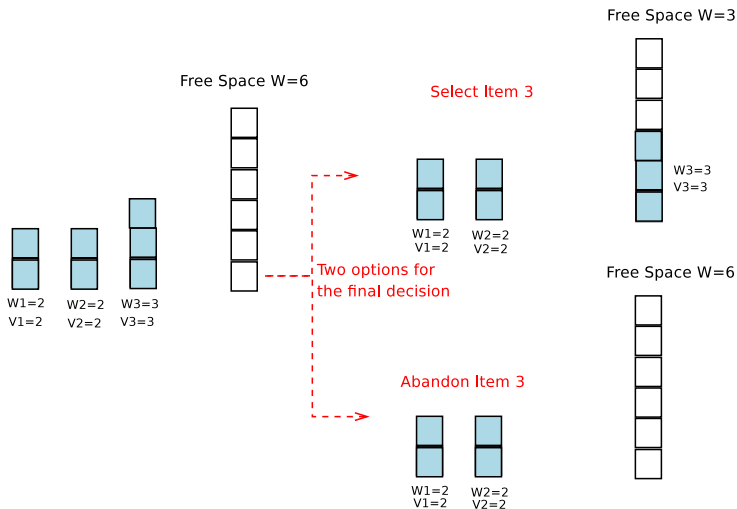
- Представим процесс решения как серию решений.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: для подмножества элементов $\{1, 2, \dots, i\}$ и ранца размером w найти наиболее ценную упаковку, обозначим ценность решения как $OPT(i, w)$.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: для подмножества элементов $\{1, 2, \dots, i\}$ и ранца размером w найти наиболее ценную упаковку, обозначим ценность решения как $OPT(i, w)$.
- Наша цель: $OPT(n, W)$.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: для подмножества элементов $\{1, 2, \dots, i\}$ и ранца размером w найти наиболее ценную упаковку, обозначим ценность решения как $OPT(i, w)$.
- Наша цель: $OPT(n, W)$.
- $OPT(n, W) = \max\{OPT(n-1, W), OPT(n-1, W - w_n) + v_n\};$



Knapsack DP1

```
1: for  $w = 1$  to  $W$  do  
2:    $M[0, w] = 0$ ;  
3: end for  
4: for  $i = 1$  to  $n$  do  
5:   for  $w = 1$  to  $W$  do  
6:      $M[i, w] = \max\{M[i - 1, w], w_i + M[i - 1, w - w_i]\}$ ;  
7:   end for  
8: end for  
9: return  $M[n, W]$ ;
```

Трудоемкость: $O(nW)$. Это псевдополиномиальное время.

INPUT: множество имеющее n элементов. Элемент i имеет вес w_i и ценность v_i . Требование к стоимости V ;

OUTPUT: выбрать подмножество предметов, чтобы минимизировать общий вес и имеющих суммарную ценность не менее V ;

Примечание: если алгоритм для этой задачи есть, то задачу о ранце можно решить с помощью этого алгоритма ,
вычислив наибольшее значение V , такое, что $OPT(n, V) \leq W$.

- Представим процесс решения как серию решений.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: вычислить наименьший размер пакета, в который упаковывается подмножество элементов из $\{1, 2, \dots, i\}$, так чтобы ценность их была не меньше V (обозначим эту ценность $OPT(i, V)$);

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: вычислить наименьший размер пакета, в который упаковывается подмножество элементов из $\{1, 2, \dots, i\}$, так чтобы ценность их была не меньше V (обозначим эту ценность $OPT(i, V)$);
- Наша цель: найти наибольшее V такое, что $OPT(n, V) \leq W$. (Отметим, что $V \leq nv^*$, где $v^* = \max_i \{v_i\}$);

- Представим процесс решения как серию решений.
- Предположим, что мы уже получили оптимальное решение S . Рассмотрим шаг решения: выбрать элемент n или отказаться от него.
- Подзадача: вычислить наименьший размер пакета, в который упаковывается подмножество элементов из $\{1, 2, \dots, i\}$, так чтобы ценность их была не меньше V (обозначим эту ценность $OPT(i, V)$);
- Наша цель: найти наибольшее V такое, что $OPT(n, V) \leq W$. (Отметим, что $V \leq nv^*$, где $v^* = \max_i \{v_i\}$);
- $OPT(n, V) =$
$$\min \begin{cases} OPT(n-1, V) & \text{отказаться от предмета } n \\ w_n & \text{загружается только предмет } n \\ w_n + OPT(n-1, V - v_n) & \text{загружается предмет } n \text{ и какие то другие} \end{cases}$$

Алгоритм динамического программирования 2

продолжение



Knapsack DP2

```
1: for  $i = 0$  to  $n$  do
2:    $M[i, 0] = 0$ ;
3: end for
4: for  $i = 1$  to  $n$  do
5:   for  $V = 1$  to  $\sum_{k=1}^i v_k$  do
6:     if  $V > \sum_{k=1}^{i-1} v_k$  then
7:        $M[i, V] = w_i + M[i - 1, V - v_i]$ ;
8:     else
9:        $M[i, V] = \min\{M[i - 1, V], w_i, w_i + M[n - 1, v - v_i]\}$ ;
10:    end if
11:  end for
12: end for
```

Трудоемкость: $O(n^2 v^*)$. Это псевдополиномиальное время.

Преобразование псевдополиномиального алгоритма в PTAS

- Идея: алгоритм хорош, когда v^* достаточно мало. Но как действовать, когда v^* является большим? Масштабировать!



		Rounding $b = 1000$		Equivalent	
V1	3278		$\bar{V1}$ 4000		$\hat{V1}$ 4
V2	1956		$\bar{V2}$ 2000		$\hat{V2}$ 2
V3	4123		$\bar{V3}$ 5000		$\hat{V3}$ 5
V4	2233		$\bar{V4}$ 3000		$\hat{V4}$ 3

Преобразование псевдополиномиального алгоритма в PTAS

- Идея: алгоритм хорош, когда v^* достаточно мало. Но как действовать, когда v^* является большим?

Масштабировать!

- Точнее, большое v_i можно уменьшить масштабированием: $\hat{v}_i = \lceil \frac{v_i}{b} \rceil$. Обозначим $\bar{v}_i = \hat{v}_i b$;

		Rounding		Equivalent			
		b = 1000					
V1	3278		$\overline{V1}$	4000		$\hat{V1}$	4
V2	1956		$\overline{V2}$	2000		$\hat{V2}$	2
V3	4123		$\overline{V3}$	5000		$\hat{V3}$	5
V4	2233		$\overline{V4}$	3000		$\hat{V4}$	3

Knapsack-Approx(ϵ)

- 1: Let $v^* = \max_i v_i$;
- 2: Let $b = \frac{\epsilon}{n} v^*$;
- 3: Установить $\hat{v}_i = \lceil \frac{v_i}{b} \rceil$ для каждого предмета i ;
- 4: Выполнить KnapsackDP2 когда ценности предметов равны \hat{v}_i , выдать оптимальное решение S ;

Алгоритм Knapsack-Approx имеет полиномиальную трудоемкость.

Действительно, $O(n^2 \hat{v}^*) = O(n^2 \frac{v^*}{b}) = O(\frac{n^3}{\epsilon})$.

b	\bar{v}	ϵ	W	# OP	Time (ms)
1	2223975	0.001	1768	889590000	18352.128
3	741325	0.010	1768	98843333	5990.893
5	444800	0.028	1768	35584000	3649.624
10	222400	0.112	1768	8896000	1836.567
30	74125	1.011	1768	988333	620.822
50	44475	2.810	1768	355800	381.982
100	22250	11.236	1768	89000	183.707
300	7425	101.010	1768	9900	60.422
500	4450	280.899	1768	3560	38.340
1000	2225	1123.6	1768	890	17.943
3000	750	10000	1809	100	6.872
5000	450	27777.8	1809	36	4.059
10000	225	111111	1809	9	3.134

Theorem

Пусть S — решение, полученное алгоритмом *Knapsack-Approx*, и S^* — допустимое решение, такое, что $\sum_{i \in S^*} w_i \leq W$. Покажем, что $\sum_{i \in S} v_i \geq \frac{1}{1+\epsilon} \sum_{i \in S^*} v_i$.

Theorem

Пусть S — решение, полученное алгоритмом *Knapsack-Approx*, и S^* — допустимое решение, такое, что $\sum_{i \in S^*} w_i \leq W$. Покажем, что $\sum_{i \in S} v_i \geq \frac{1}{1+\epsilon} \sum_{i \in S^*} v_i$.

Доказательство.

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \bar{v}_i \quad (\text{поскольку } v_i \leq \bar{v}_i) \quad (8)$$

$$\leq \sum_{i \in S} \bar{v}_i \quad (S \text{ — оптимальное решение}) \quad (9)$$

$$\leq \sum_{i \in S} (v_i + b) \quad (\text{поскольку } \bar{v}_i \leq v_i + b) \quad (10)$$

$$\leq nb + \sum_{i \in S} v_i \quad (11)$$

$$\leq (1 + \epsilon) \sum_{i \in S} v_i \quad (\text{так как } nb \leq \epsilon \sum_{i \in S} v_i) \quad (12)$$



Почему b полагаем равным $b = \frac{\epsilon}{n} v^*$?

Заметим: b полагаем равным $b = \frac{\epsilon}{n} v^*$ по двум соображениям:

- ❶ Трудоемкость: $O(n^2 \frac{v^*}{b})$ является полиномом от n и $\frac{1}{\epsilon}$.
- ❷ Степень приближения: $nb \leq \epsilon \sum_{i \in S} v_i$.

Параметрическая обрезка

Алгоритм состоит из трех шагов:

- 1 Обрезка: Предположим, у нас есть предположение о значении OPT , обозначим эту оценку как t и рассмотрим ее как параметр. Для каждого заданного t будем упрощать исходную задачу путем удаления предметов, которые не будут использоваться ни в одном решении со стоимостью $> t$. Обозначим сокращенный экземпляр задачи как $I(t)$
- 2 Нижняя граница: семейство задач $I(t)$ используется для вычисления нижней границы OPT , обозначим ее t^* ;
- 3 Хорошее решение: это решение задачи $I(\alpha t^*)$ со специально выбранным α .