Operations Research (Master's Degree Course)

9. Relaxations

Silvano Martello
DEI "Guglielmo Marconi", Università di Bologna, Italy

Branch-and-bound algorithms

- $\mathcal{NP}$-hard problem $(z, F) =$ find $y^* \in F$ such that $z(y^*) \geq z(y)\ \forall y \in F$.
- Ingredients of a branch-and-bound algorithm:
  - **branch-decision tree**; $\surd$
  - **relaxations** (bounds);
  - **reduction techniques** (preprocessing);
  - **approximation algorithms**.
- Branch-and-bound method:
  - $P^0 = (z, F(P^0))$ (problem to be solved): $Z(P^0) = \max\{z(y) : y \in F(P^0)\}$;
  - **subdivide** P^0 into subproblems $P^1, P^2, \ldots, P^{n_0}$ that "represent" P^0:
    - **subdivide** $F(P^0)$ into $F(P^1), F(P^2), \ldots, F(P^{n_0})$ st $\cup_{k=1}^{n_0} F(P^k) = F(P^0)$:
    - $Z(P^k) = \max\{z(y) : y \in F(P^k)\}$
      $\Rightarrow Z(P^0) = \max\{Z(P^1), Z(P^2), \ldots, Z(P^{n_0})\}$, i.e.,
  - (Solve P^0) $\Leftrightarrow$ (*solve* P^k $(k = 1, \ldots, n_0)$), where ***solve*** can mean

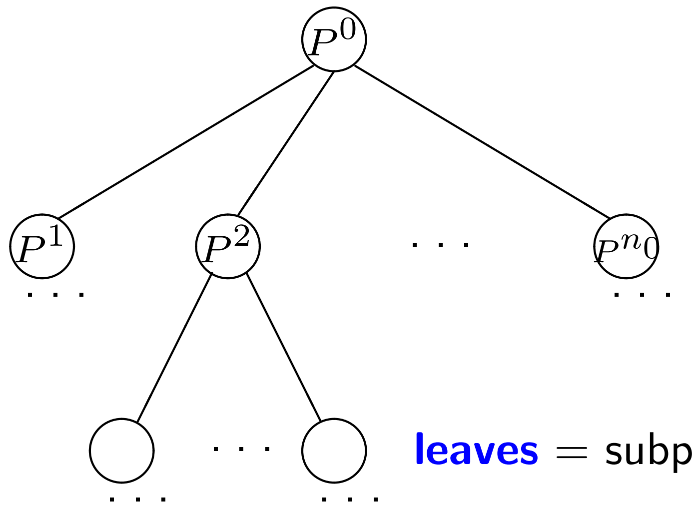
    1. find the optimal solution to P^k, **or**

    2. prove that $F(P^k) = \emptyset$ **or**

    3. prove that $Z(P^k) \leq Z$ ($Z =$ incumbent solution value).

Branch-and-bound algorithms (cont'd)

- subdivision iterated for each subproblem that cannot be ***solved***:



- For each P^k, compute $U(P^k)$ **(upper bound)** such that $U(P^k) \geq Z(P^k)$

  $\Rightarrow$ **3.** check that $U(P^k) \leq Z$.

- Bound for $P^k = (z, F(P^k))$, $\Leftarrow$ **relaxation** of P^k: problem $R(P^k) = (z_r, F_r(P^k))$ s.t.

  a. $F_r(P^k) \supseteq F(P^k)$,

  b. $z_r(y) \geq z(y) \; \forall y \in F(P^k)$.

- $U(P^k) = Z(R(P^k)) = \max\{z_r(y) : y \in F_r(P^k)\}$.

- **Criteria:** $Z(R(P^k))$ as close as possible to $Z(P^k)$ (as low as possible);

  $R(P^k)$ sufficiently "easy" to solve.

Problem to relax (LP01, but results extend to ILP)

$$Z(P) = \max \sum_{j=1}^{n} v_j x_j \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i = 1, \ldots, m) \tag{2}$$

$$\sum_{j=1}^{n} d_{kj} x_j = e_k \quad (k = 1, \ldots, l) \tag{3}$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)\,. \tag{4}$$

Relaxation by constraint elimination

- The simplest one: eliminate one or more constraints to obtain a well-structured problem, e.g.,

$$Z(E(P)) = \max \sum_{j=1}^{n} v_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i = 1, \ldots, m)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)\,,$$

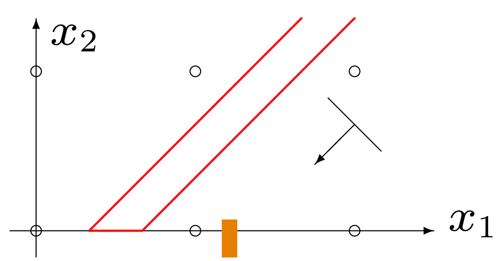
- *Multi-dimensional 0-1 knapsack problem*: m "weights" per item, m capacities.
- If the optimal solution x^* to $E(P)$ is **feasible for P** then it is **optimal for P**.

Continuous relaxation

- Replace constraints $x_j \in \{0, 1\}$ with

$$0 \le x_j \le 1 \qquad (j = 1, \dots, n)$$

  obtaining LP, hence an upper bound $Z(C(P))$.
- If coefficients v_j are integer, improved upper bound $\lfloor Z(C(P)) \rfloor$.
- If the optimal solution x^* to $C(P)$ is **feasible for P** then it is **optimal for P**.
- For a general relaxation R:
  1. if $R(P)$ is impossible then P is impossible;
  2. if $R(P)$ is unbounded, nothing is known on P.
- For the continuous relaxation $C(P)$,
  2. if $C(P)$ is unbounded then P is either unbounded or impossible. Remind:



Surrogate relaxation

1. Select a set of constraints (e.g., (2)) and add a redundant constraint to P:

$$\sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} a_{ij} x_j \le \sum_{i=1}^{m} \pi_i b_i,$$

with $\pi_i \ge 0 \; \forall i$ **surrogate multipliers**. Problem unchanged.

2. Eliminate the selected constraints. **Surrogate relaxation**:

$$\begin{aligned} Z(S(P,\pi)) = \quad \max \quad & \sum_{j=1}^{n} v_j x_j \\ & \sum_{j=1}^{n} \widehat{a}_j x_j \le \widehat{b}, \\ & \sum_{j=1}^{n} d_{kj} x_j = e_k \;\; (k = 1, \dots, l) \\ & x_j \in \{0,1\} \quad (j = 1, \dots, n) \end{aligned}$$

where $\widehat{a}_j = \sum_{i=1}^{m} \pi_i a_{ij} \;\; (j = 1, \dots, n)$ and $\widehat{b} = \sum_{i=1}^{m} \pi_i b_i$.

- If the optimal solution x^* to $S(P)$ is **feasible for P** then it is **optimal for P**.
- $Z(S(P,\pi))$ is a valid upper bound for any vector π. **Surrogate dual problem**:

$$Z(S(P,\pi^*)) = \min_{\pi \ge 0} \{ Z(S(P,\pi)) \} .$$

Lagrangian relaxation of inequalities

1. Select a set of constraints (e.g., (2)) and add a non-negative term to the objective function:

$$\max \sum_{j=1}^{n} v_j x_j + \sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j),$$

with $\lambda_i \geq 0 \; \forall i$ **Lagrangian multipliers**: valid upper bound.

2. Eliminate the selected constraints. **Lagrangian relaxation**:

$$Z(L(P, \lambda)) = \sum_{i=1}^{m} \lambda_i b_i + \max \sum_{j=1}^{n} \widehat{v}_j x_j$$

$$\sum_{j=1}^{n} d_{kj} x_j = e_k \;\; (k = 1, \dots, l)$$

$$x_j \in \{0, 1\} \quad (j = 1, \dots, n),$$

where $\widehat{v}_j = v_j - \sum_{i=1}^{m} \lambda_i a_{ij} \;\; (j = 1, \dots, n)$.

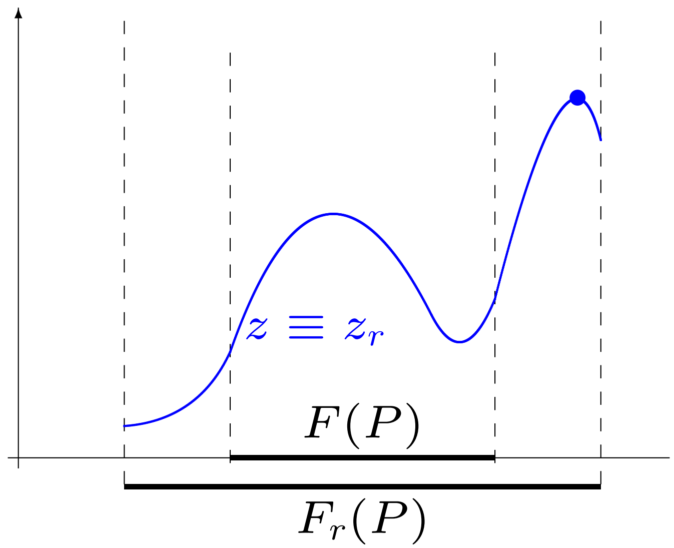
- If the optimal solution x^* to $L(P)$ is **feasible for** P then it is **not** necessarily **optimal for** P:

  it is **optimal** if the two objective functions have the same value, i.e., if

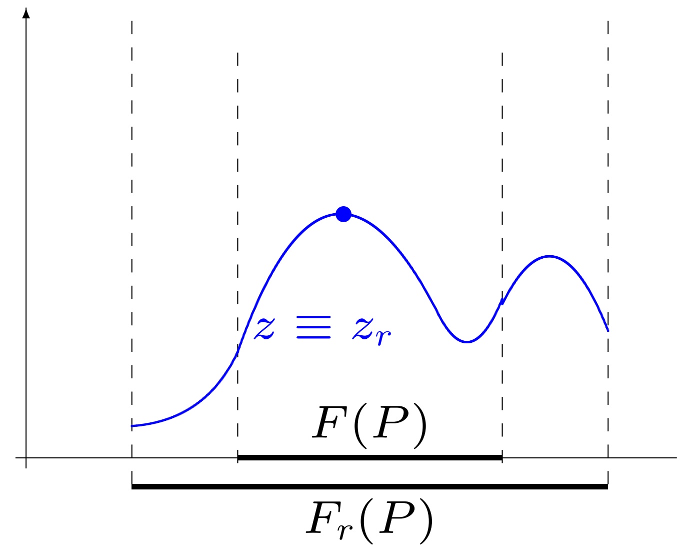
$$\sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j^*) = 0$$ ($\Leftrightarrow$ if, $\forall \; i$, either λ_i is 0 or constraint i is tight).

- **Lagrangian dual problem**: find λ^* such that $Z(L(P, \lambda^*)) = \min_{\lambda \geq 0} \{Z(L(P, \lambda))\}$.

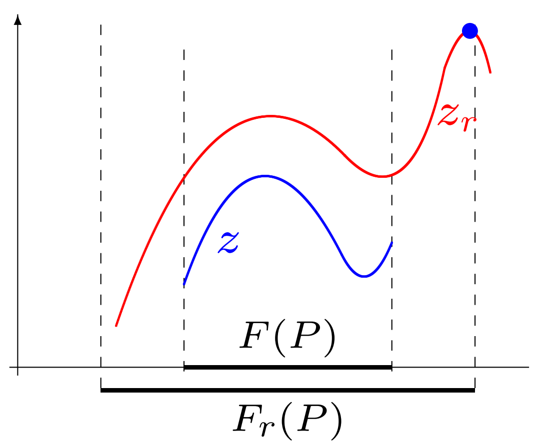
Different kinds of relaxation



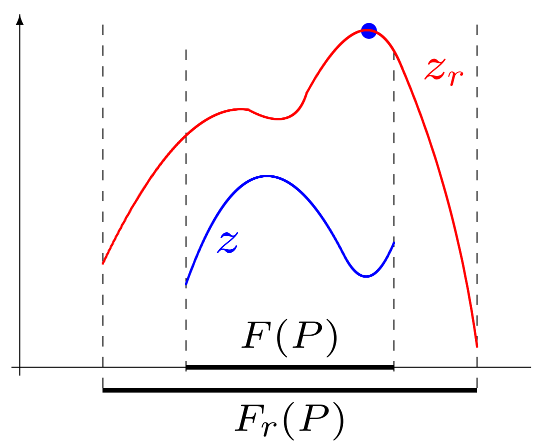
unfeasible solution



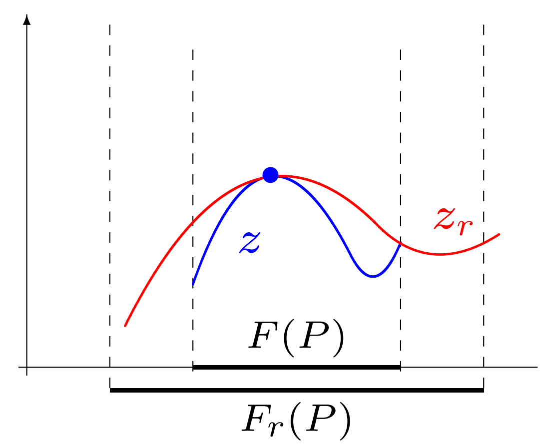
feasible and optimal solution



unfeasible solution



feasible non optimal solution



feasible and optimal solution

Lagrangian relaxation of equations

1. Select a set of constraints (e.g., (3)) and add a nul term to the objective function:

$$\max \sum_{j=1}^{n} v_j x_j + \sum_{k=1}^{l} \lambda_k (e_k - \sum_{j=1}^{n} d_{kj} x_j),$$

with $\lambda_k \gtrless 0 \; \forall k$ **Lagrangian multipliers**. Problem unchanged.

2. Eliminate the selected constraints. **Lagrangian relaxation**:

$$Z(L(P,\lambda)) = \sum_{k=1}^{l} \lambda_k e_k + \quad \max \sum_{j=1}^{n} \widehat{v}_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i = 1, \dots, m)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \dots, n)\,.$$

where $\widehat{v}_j = v_j - \sum_{k=1}^{l} \lambda_k d_{kj}$.

- If the optimal solution x^* to $L(P)$ is **feasible for P** then it is **optimal for P**.
- **Lagrangian dual problem**: find λ^* such that $Z(L(P,\lambda^*)) = \min_\lambda\{Z(L(P,\lambda))\}$.

Lagrangian decomposition

1. Select a set of constraints (e.g., (3)) and express it using new variables y.
Select a part of the objective function and express it using new variables y.
Weight the two parts with α and β such that $\alpha + \beta = 1$.
Add a constraint imposing $x = y$. Problem unchanged:

$$
\begin{aligned}
\max \quad & \alpha \sum_{j=1}^{n} v_j x_j + \beta \sum_{j=1}^{n} v_j y_j \\
& \sum_{j=1}^{n} a_{ij} x_j \leq b_i && (i = 1, \dots, m) \\
& \sum_{j=1}^{n} d_{kj} y_j = e_k && (k = 1, \dots, l) \\
& x_j \in \{0, 1\} && (j = 1, \dots, n) \\
& y_j \in \{0, 1\} && (j = 1, \dots, n) \\
& y_j = x_j && (j = 1, \dots, n)\,.
\end{aligned}
$$

Lagrangian decomposition (cont'd)

2. Lagrangian relaxation of the new constraints. **Decomposition**:

$$
\begin{aligned}
Z(D(P,\lambda)) = \quad \max \quad & \sum_{j=1}^{n}(\alpha v_j + \lambda_j)x_j + \sum_{j=1}^{n}(\beta v_j - \lambda_j)y_j \\
& \sum_{j=1}^{n} a_{ij}x_j \le b_i \quad (i = 1,\dots,m) \\
& \sum_{j=1}^{n} d_{kj}y_j = e_k \quad (k = 1,\dots,l) \\
& x_j \in \{0,1\} \qquad (j = 1,\dots,n) \\
& y_j \in \{0,1\} \qquad (j = 1,\dots,n)\,.
\end{aligned}
$$

- $DX(P,\lambda)$ = problem in x;
- $DY(P,\lambda)$ = problem in y;
- upper bound $Z(D(P,\lambda)) = Z(DX(P,\lambda)) + Z(DY(P,\lambda))$.
- If the optimal solution x^* to $D(P,\lambda)$ is **feasible for** P then it is **optimal for** P.
- **Lagrangian dual problem**: find λ^* such that $Z(D(P,\lambda^*)) = \min_\lambda\{Z(D(P,\lambda))\}$.

Subgradient Optimization

For well-structured problems it is sometimes possible to determine the **optimal multipliers** through theoretical analysis.

For general cases **good multipliers** are produced by the **subgradient method**. We consider the case of Lagrangian relaxation of inequalities:

- objective function: $\max \sum_{j=1}^{n} v_j x_j + \sum_{i=1}^{m} \lambda_i L_i$

  where *subgradient* $L_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$ is the slack in the ith relaxed constraint:

  - if $L_i < 0$ (violated constraint) the term $\lambda_i L_i$ penalizes the objective function;
  - if $L_i > 0$ (loose constraint) the term $\lambda_i L_i$ rewards the objective function.

- **Algorithm:**

  start with any λ, and a prefixed initial step $t > 0$;
  while halting condition is not satisfied **do**
      begin
          solve $L(P, \lambda)$;
          for $i := 1$ **to** m **do** $\lambda_i := \max(0,\ \lambda_i - tL_i)$;
          correct t;
      end

Subgradient Optimization (cont'd)

- **Intuitively:**

  if $L_i < 0$ then λ_i is too small and must increase;

  if $L_i > 0$ then λ_i is too large and must decrease;

  if $L_i = 0$ then λ_i is fine (optimality condition).

- Standard formula for the **step updating**:

$$t = \vartheta \frac{Z(L(P,\lambda)) - Z}{\sum_{i=1}^{m} L_i^2}, \text{ where}$$

  Z = incumbent solution value;

  ϑ decreases with the number of iterations, e.g.,:

    start with $\vartheta = 2$; halve ϑ after some iterations with no improvement.

- **Two halting conditions:**

  1. $\forall \; i$, either $(L_i = 0)$ or $(L_i > 0 \textbf{ and } \lambda_i = 0)$ (optimal multipliers);

  2. maximum number of iterations reached.

- **Non-monotone** method: the best λ value is stored.

- Better methods are much more complicated.

- Non-satisfactory extension to surrogate relaxations.

Dominance among relaxations

- Notation: $R_{(\mathsf{v})}(P)$ = relaxation R of constraints (v).
- **Property** *Both the dual surrogate and the Lagrangian relaxation dominate constr. elimination.*

  Proof $E_{(\mathsf{v})}(P) = S_{(\mathsf{v})}(P, \pi)$ with $\pi = 0$. $E_{(\mathsf{v})}(P) = L_{(\mathsf{v})}(P, \lambda)$ with $\lambda = 0$. $\square$
- **Lagrangian vs Surrogate:**

$$
\begin{aligned}
Z(S_{(2)}(P,\mu)) = \quad \max \quad & \sum_{j=1}^{n} v_j x_j \\
& \sum_{i=1}^{m} \mu_i \sum_{j=1}^{n} a_{ij} x_j \le \sum_{i=1}^{m} \mu_i b_i \qquad (10) \\
& \sum_{j=1}^{n} d_{kj} x_j = e_k \quad (k = 1, \dots, l) \\
& x_j \in \{0,1\} \qquad (j = 1, \dots, n).
\end{aligned}
$$

$$
\begin{aligned}
Z(L_{(2)}(P,\mu)) = \quad \max \quad & \sum_{j=1}^{n} v_j x_j + \sum_{i=1}^{m} \mu_i b_i - \sum_{i=1}^{m} \mu_i \sum_{j=1}^{n} a_{ij} x_j \\
& \sum_{j=1}^{n} d_{kj} x_j = e_k \quad (k = 1, \dots, l) \\
& x_j \in \{0,1\} \qquad (j = 1, \dots, n)\,.
\end{aligned}
$$

Hence: $L_{(2)}(P,\mu) = L_{(10)}(S_{(2)}(P,\mu), 1)$, i.e.,

Dominance among relaxations (cont'd)

- **Property** *The dual surrogate relaxation dominates the dual Lagrangian relaxation.*

- **Property** *The dual Lagrangian decomposition dominates the separate dual Lagrangian relaxations*, i.e.,

$$Z(D_{(\mathsf{v}),(\mathsf{w})}(P,\lambda^*)) \leq \min\left(\ Z(L_{(\mathsf{v})}(P,\mu^*)),\ Z(L_{(\mathsf{w})}(P,\nu^*))\ \right)\ .$$

  Proof Omitted.

- **Property** *The dual Lagrangian relaxation of any set of constraints dominates the continuous relaxation*, i.e.,

$$Z(L(P,\lambda^*)) \leq Z(C(P)).$$

  Proof Omitted.

- **Summing up:**

$$\left[\begin{array}{c} Z(S_{(\mathsf{v})}(P,\pi^*)) \\ Z(D_{(\mathsf{v}),(\mathsf{w})}(P,\lambda^*)) \end{array}\right] \leq Z(L_{(\mathsf{v})}(P,\lambda^*)) \leq \left[\begin{array}{c} Z(C(P)) \\ Z(E_{(\mathsf{v})}(P)) \end{array}\right].$$

Integrality property

- **Definition** An ILP has the **integrality property** if, for any instance, its continuous relaxation has an optimal integer solution.

- **Property** *If a Lagrangian relaxation has the integrality property, then the value of the dual Lagrangian relaxation is equal to the value of the continuous relaxation.*

  Proof If a Lagrangian relaxation has integrality property then

  $$Z(L(P,\lambda)) = Z(C(L(P,\lambda))) \ \forall \lambda \, .$$

  The feasible region of $C(L(P,\lambda))$ contains that of $C(P)$, so

  $$Z(C(L(P,\lambda))) \geq Z(C(P)) \ \forall \lambda \, .$$

  It follows $Z(L(P,\lambda)) \geq Z(C(P)) \ \forall \lambda$, hence $Z(L(P,\lambda^*)) = Z(C(P))$. $\square$

- **Property** *If a surrogate relaxation has the integrality property, then the value of the dual surrogate relaxation is equal to the value of the continuous relaxation.*

  Proof Very similar. $\square$

Relaxations of the 0-1 (single) Knapsack Problem KP01

- KP01: $\max Z(\text{KP01}) = \sum_{j=1}^{n} p_j x_j : \sum_{j=1}^{n} w_j x_j \le c,\ x_j \in \{0,1\}\ (j = 1, \dots, n)$.
- **Continuous relaxation:** Dantzig's bound $U = \lfloor z(C(\text{KP01}) \rfloor)$.
- **Constraint elimination:** trivial upper bound $\sum_{j=1}^{n} p_j$.
- **Surrogate relaxation:** impossible.
- **Lagrangian relaxation:** single multiplier $\lambda \ge 0$:

$$Z(L(\text{KP01}, \lambda) = \max \sum_{j=1}^{n} p_j x_j + \lambda(c - \sum_{j=1}^{n} w_j x_j) = \lambda c + \quad \max \quad \sum_{j=1}^{n} \widehat{p}_j x_j$$

$$x_j \in \{0,1\} \quad \forall\ j,$$

$$(\widehat{p}_j = p_j - \lambda w_j).\ x_j = \begin{cases} 1 \text{ if } \widehat{p}_j > 0\,; \\ 0 \text{ if } \widehat{p}_j \le 0\,, \end{cases} \Rightarrow z(L(\text{KP01}, \lambda)) = \lambda c + \sum_{p_j/w_j > \lambda} \widehat{p}_j.$$

- $L(\text{KP01}, \lambda)$ has the integrality property, hence $Z(L(\text{KP01}, \lambda^*)) = Z(C(\text{KP01})) = U$.
- We show that $\lambda^* = p_s/w_s$ (s, the critical item):

$$z(L(\text{KP01}, \lambda^*)) = c\,\frac{p_s}{w_s} + \sum_{p_j/w_j > p_s/w_s} \left(p_j - \frac{p_s}{w_s} w_j\right)$$

$\Rightarrow$ for sorted elements $z(L(\text{KP01}, \lambda^*)) = \sum_{j=1}^{s-1} p_j + \frac{p_s}{w_s}(c - \sum_{j=1}^{s-1} w_j) = z(C(K))$.

0-1 Multiple Knapsack Problem

- m containers of capacities c_i $(i = 1, \dots, m)$.
- select m disjoint feasible subsets of elements that produce the maximum total profit.

$$x_{ij} = \begin{cases} 1 \text{ if element } j \text{ is assigned to container } i\,; \\ 0 \text{ otherwise;} \end{cases}$$

$$\begin{array}{llll} \text{(MKP01)} \max \displaystyle\sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} & & & \\ \displaystyle\sum_{j=1}^{n} w_j x_{ij} & \leq & c_i & (i \in M = \{1, \dots, m\}) \\ \displaystyle\sum_{i=1}^{m} x_{ij} & \leq & 1 & (j \in N = \{1, \dots, n\}) \\ x_{ij} & \in & \{0, 1\} & (i \in M,\ j \in N)\,. \end{array}$$

- Generalization of KP01 $\Rightarrow$ $\mathcal{NP}$-hard. (It can be proved to be strongly $\mathcal{NP}$-hard.)
- Profits, weights and capacities positive integers; items sorted by non-increasing p_j/w_j;
- $w_j \leq \max_{i \in M}\{c_i\}\ \ \forall j \in N$; $c_i \geq \min_{j \in N}\{w_j\}\ \ \forall i \in M$; $\sum_{j \in N} w_j > c_i\ \ \forall i \in M$.

Surrogate relaxation of MKP01

- m multipliers $\pi_i \geq 0$:

$$
\begin{aligned}
\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} & & & \\
\sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} w_j x_{ij} & \leq & \sum_{i=1}^{m} \pi_i c_i & \\
\sum_{i=1}^{m} x_{ij} & \leq & 1 & \quad (j \in N) \\
x_{ij} & \in & \{0,1\} & \quad (i \in M,\ j \in N)\,.
\end{aligned}
$$

- **Observation:** To obtain a non-trivial bound, it must be $\pi_i > 0 \ \ \forall\ i$.

  Proof If $\exists\ \pi_k = 0$ then an optimal solution is $x_{kj} = 1\ \forall\ j$
  $\Rightarrow$ bound $= \sum_{j=1}^{n} p_j$. $\square$

Surrogate relaxation of MKP01

- **Property:** the optimal dual multipliers are $\pi_i^* = Q \ \ \forall\ i$ (Q any positive constant).

  Proof Let (x_{ij}^*) be the optimal solution to the relaxation. Let $k = \arg\min \pi_i$.

  equivalent solution: for each j such that $x_{ij}^* = 1$ and $i \neq k$, set $x_{ij}^* = 0$ and $x_{kj}^* = 1$.

  Hence the relaxed problem is equivalent to the KP01:

$$\max \sum_{j=1}^{n} p_j x_{kj}$$

$$\begin{aligned}\sum_{j=1}^{n} w_j x_{kj} \quad &\leq \quad \left\lfloor \sum_{i=1}^{m} \pi_i c_i / \pi_k \right\rfloor \\ x_{kj} \quad &\in \quad \{0,1\} \ \ (j \in N).\end{aligned}$$

  Setting $\pi_i = Q > 0 \ \ \forall\ i$ we get the smallest capacity hence the lowest upper bound. $\square$

- The surrogate relaxation of MKP01 is thus equivalent to

$$\max \sum_{j=1}^{n} p_j y_j$$

$$\begin{aligned}\sum_{j=1}^{n} w_j y_j \quad &\leq \quad \sum_{i=1}^{m} c_i \\ y_j \quad &\in \quad \{0,1\} \quad (j \in N),\end{aligned}$$

Lagrangian relaxation of MKP01

- n multipliers $\lambda_j \geq 0$:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} + \sum_{j=1}^{n} \lambda_j (1 - \sum_{i=1}^{m} x_{ij})$$

$$= \sum_{j=1}^{n} \lambda_j + \max \sum_{i=1}^{m} \sum_{j=1}^{n} \widehat{p}_j x_{ij}$$

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i \quad (i \in M)$$

$$x_{ij} \in \{0,1\} \quad (i \in M, j \in N)$$

with $\widehat{p}_j = p_j - \lambda_j$. Relaxation: an item can be inserted into more containers.

- Equivalent to m independent KP01s of the form, for $i = 1, \ldots, m$,

$$z_i = \max \sum_{j=1}^{n} \widehat{p}_j x_{ij}$$

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i$$

$$x_{ij} \in \{0,1\} (j \in N)$$

(same profits and weights, different capacities). Upper bound $= \sum_{j=1}^{n} \lambda_j + \sum_{i=1}^{m} z_i$.

- $\exists$ multipliers $\overline{\lambda}$ st $Z(C(L(\text{MKP01}, \overline{\lambda}))) = Z(C(S(\text{MKP01})) = Z(C(\text{MKP01}))$.

- **Example:** $n = 6$, $m = 2$,
$$\begin{array}{rcl} (p_j) & = & (\ 110,\ 150,\ 70,\ 80,\ 30,\ 5\ ) \\ (w_j) & = & (\ 40,\ 60,\ 30,\ 40,\ 20,\ 5\ ) \\ (c_i) & = & (\ 65,\ 85) \end{array}$$
- **Surrogate relaxation:** KP01 of capacity $c = 150$.
- Optimal solution ($\mathcal{NP}$-hard) problem) $(y_j^*) = (\ 1,\ 1,\ 1,\ 0,\ 1,\ 0) \Rightarrow$ Upper bound = 360.
- Using an upper bound: $U = 370$ (Dantzig), or $\overline{U} = 363$ (improved bound).
- Deciding if a surrogate relaxation solution (y_j^*) is feasible for the problem is $\mathcal{NP}$-complete.
- **Lagrangian relaxation:** We use multipliers
$$\overline{\lambda}_j = \begin{cases} p_j - w_j \dfrac{p_s}{w_s} & \text{se} \quad j < s\,; \\ 0 & \text{se} \quad j \geq s\,, \end{cases}$$
with s = critical item of the surrogate relaxation ($s = 4$):
$(\overline{\lambda}_j) = (30,\ 30,\ 10,\ 0,\ 0,\ 0) \Rightarrow (\widehat{p}_j) = (80,\ 120,\ 60,\ 80,\ 30,\ 5)$.
- Optimal solution to the two resulting KP01s:
$$(x_{ij}^*) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$
$\Rightarrow$ Upper bound = 125 + 165 + 70 = 360.
- Deciding if a Lagrangian relaxation solution (x_{ij}^*) is feasible for the original problem is easy.

Reduction techniques

- **Reduction:** method to preventively establish the optimal value of some variables.
- Z = value of a feasible solution to P; define, for $h \in \{1, \dots, n\}$, problem P if $x_h = 1$:

$$Z(P_{[x_h=1]}) = v_h + \quad \max \sum_{\substack{j=1 \\ j \neq h}}^{n} v_j x_j$$

$$\sum_{\substack{j=1 \\ j \neq h}}^{n} a_{ij} x_j \leq b_i - a_{ih} \quad (i = 1, \dots, m)$$

$$\sum_{\substack{j=1 \\ j \neq h}}^{n} d_{kj} x_j = e_k - d_{kh} \quad (k = 1, \dots, l)$$

$$x_j \in \{0, 1\} \quad (j = 1, \dots, n; j \neq h);$$

- Compute $U(P_{[x_h=1]})$ = upper bound for $Z(P_{[x_h=1]})$. Then
- *If $U(P_{[x_h=1]}) \leq Z$, it must be $x_h = 0$ in any solution to P of value $> Z$.*
- By similarly defining $P_{[x_h=0]}$ (problem P if $x_h = 0$):
- *If $U(P_{[x_h=0]}) \leq Z$, it must be $x_h = 1$ in any solution to P of value $> Z$.*

Reduction techniques (cont'd)

- We are only interested in solutions of value $> Z$, so

  compute $U(P_{[x_h=1]})$ and $U(P_{[x_h=0]})$ for $h = 1, \ldots, n$, and define:

$$\begin{aligned} J_0 &= \{h : U(P_{[x_h=1]}) \leq Z\}\,; \\ J_1 &= \{h : U(P_{[x_h=0]}) \leq Z\}\,; \\ J &= \{1, \ldots, n\} \backslash (J_0 \cup J_1)\,. \end{aligned}$$

- **Special case 1: if** $J_0 \cap J_1 \neq \emptyset$ **then** *the solution of value Z is optimal for P.*
- **Special case 2: if** $J = \emptyset$ **then** *we have a solution of value* $\sum_{h \in J_1} v_h$, which however
  - is not necessarily feasible (J_0 and J_1 come from relaxations);
  - is not necessarily better than Z (J_0 and J_1 come from necessary but non sufficient conditions), so

  if $\sum_{h \in J_1} a_{ih} \leq b_i \ \forall\ i$ *and* $\sum_{h \in J_1} d_{kh} = e_k \ \forall\ k$, **then** *the optimal solution to P is the best one between*
  - *the one induced by J_0 and J_1 (of value $\sum_{h \in J_1} v_h$), and*
  - *the solution of value Z*

  else *the solution of value Z is optimal for P.*

Reduction techniques (cont'd)

- **General case: if $J_0 \cap J_1 = \emptyset$ and $J \neq \emptyset$, then**
- we solve the **reduced problem**

$$
\begin{aligned}
Z(P_{[J_0,J_1]}) = \quad \max \quad & \sum_{j \in J} v_j x_j \\
& \sum_{j \in J} a_{ij} x_j \leq b_i - \sum_{h \in J_1} a_{ih} \quad (i = 1, \ldots, m), \\
& \sum_{j \in J} d_{kj} x_j = e_k - \sum_{h \in J_1} d_{kh} \quad (k = 1, \ldots, l), \\
& x_j \in \{0, 1\} \qquad\qquad\qquad (j \in J).
\end{aligned}
$$

Again, $P_{[J_0,J_1]}$ is not necessarily feasible or better than the solution of value Z, so

if $P_{[J_0,J_1]}$ *is infeasible* **then** *the solution of value Z is optimal for P*

else *let* $J^* = \{j \in J : x_j = 1$ *in the optimal solution to*$P_{[J_0,J_1]}\}$:

the optimal solution to P is the best one between

– *the one induced by J_0, J_1 and J^* (of value $\sum_{h \in J_1} v_h + \sum_{j \in J^*} v_j$), and*

– *the solution of value Z.*

Reduction algorithm for KP01

- U = Dantzig's bound for the problem;
- U_j^1 (resp. U_j^0)= upper bound if $x_j = 1$ (resp $x_j = 0$);
- for $j = 1, \ldots, s-1$ we have $U_j^1 = U$; for $j = s+1, \ldots, n$ we have $U_j^0 = U$.

procedure Reduce KP01:
begin
  $Z :=$ value of a feasible solution; $U :=$ Dantzig's upper bound;
  if $Z = U$ **then stop** (**comment:** Z is optimal);
  for $j := 1$ **to** s **do** compute U_j^0;
  for $j := s$ **to** n **do** compute U_j^1;
  $J_0 := \{j \geq s : U_j^1 \leq Z\}$, $J_1 := \{j \leq s : U_j^0 \leq Z\}$;
  if $J_0 \cap J_1 = \{s\}$ **or** $\sum_{j \in J_1} w_j > c$ **then stop** (**comment:** Z is optimal);
  else
    begin
      $J := \{1, \ldots, n\} \setminus (J_0 \cup J_1)$;
      if $J = \emptyset$ **then stop** (**comment:** solution $= \max(Z, \sum_{j \in J_1} p_j)$);
      else solve(*) the reduced problem

$$z = \sum_{j \in J_1} p_j + \max \sum_{j \in J} p_j x_j : \sum_{j \in J} w_j x_j \leq c - \sum_{j \in J_1} w_j, x_j \in \{0,1\} (j \in J)$$

      and determine the optimal solution, of value $\max(z, Z)$
    end
end.

- **Time complexity** (excluding (*)): $O(n^2)$ ($O(n \log n)$ using special techniques).

Reduction algorithm for KP01 (cont'd)

- **Example:**
$$\begin{array}{rcl}(p_j) &=& (\ 70,\ 20,\ 39,\ 36,\ 15,\ 5,\ 10\ )\\ (w_j) &=& (\ 31,\ 10,\ 20,\ 19,\ 8,\ 3,\ 6\ )\\ c &=& 50\end{array}$$
- Feasible solution $x = (1, 1, 0, 0, 1, 0, 0)$, of value $Z = 105$.
- Upper bound: $s = 3$, $\bar{c} = c - \sum_{j=1}^{s-1} w_j = 9$, $\bar{p} = \sum_{j=1}^{s-1} p_j = 90$, $U = 107$.

  $j = 1$: $s' = 5$, $\bar{c}' = 1$, $\bar{p}' = 95$, $U_1^0 = 96 \leq Z \Rightarrow J_1 = \{1\}$;

  $j = 2$: $s' = 3$, $\bar{c}' = 19$, $\bar{p}' = 70$, $U_2^0 = 107 > Z$;

  $j = 3$: $s' = 4$, $\bar{c}' = 9$, $\bar{p}' = 90$, $U_3^0 = 107 > Z$;

  $j = 3$: $s' = 1$, $\bar{c}' = 30$, $\bar{p}' = 39$, $U_3^1 = 106 > Z$;

  $j = 4$: $s' = 2$, $\bar{c}' = 0$, $\bar{p}' = 106$, $U_4^1 = 106 > Z$;

  $j = 5$: $s' = 3$, $\bar{c}' = 1$, $\bar{p}' = 105$, $U_5^1 = 106 > Z$;

  $j = 6$: $s' = 3$, $\bar{c}' = 6$, $\bar{p}' = 95$, $U_6^1 = 106 > Z$;

  $j = 7$: $s' = 3$, $\bar{c}' = 3$, $\bar{p}' = 100$, $U_7^1 = 105 \leq Z \Rightarrow J_0 = \{7\}$.
- Reduced problem: $J = \{2, 3, 4, 5, 6\}$, $\bar{c} = 19$; solution $x_4 = 1$, $x_2 = x_3 = x_5 = x_6 = 0$.

  Overall solution $x^* = (1, 0, 0, 1, 0, 0, 0)$, $z = 36 + 70 = 106$.
- For each j, $\bar{p}'$ is the value of a feasible solution. $\Rightarrow$ **At each iteration we could execute**

  if $\bar{p}' > Z$ **then** $Z := \bar{p}'$ (and store the solution vector)
- At iteration $j = 4$ we find $Z := 106$ ($x = (1, 0, 0, 1, 0, 0, 0)$). $\Rightarrow J_0 = \{4, 5, 6, 7\}$;
- Reduced problem: $J = \{2, 3\}$, $\bar{c} = 19$; $x_2 = 1$, $x_3 = 0$, $z = 20 + 70 = 90 < Z$ (Z is optimal).