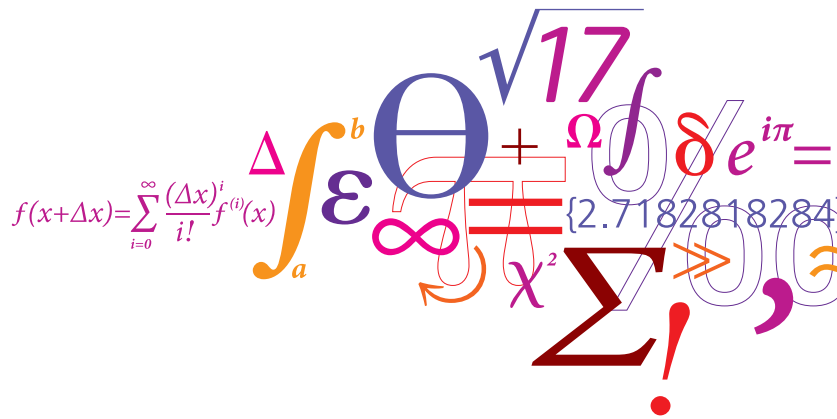
Master Problem Issues

Jesper Larsen[1]

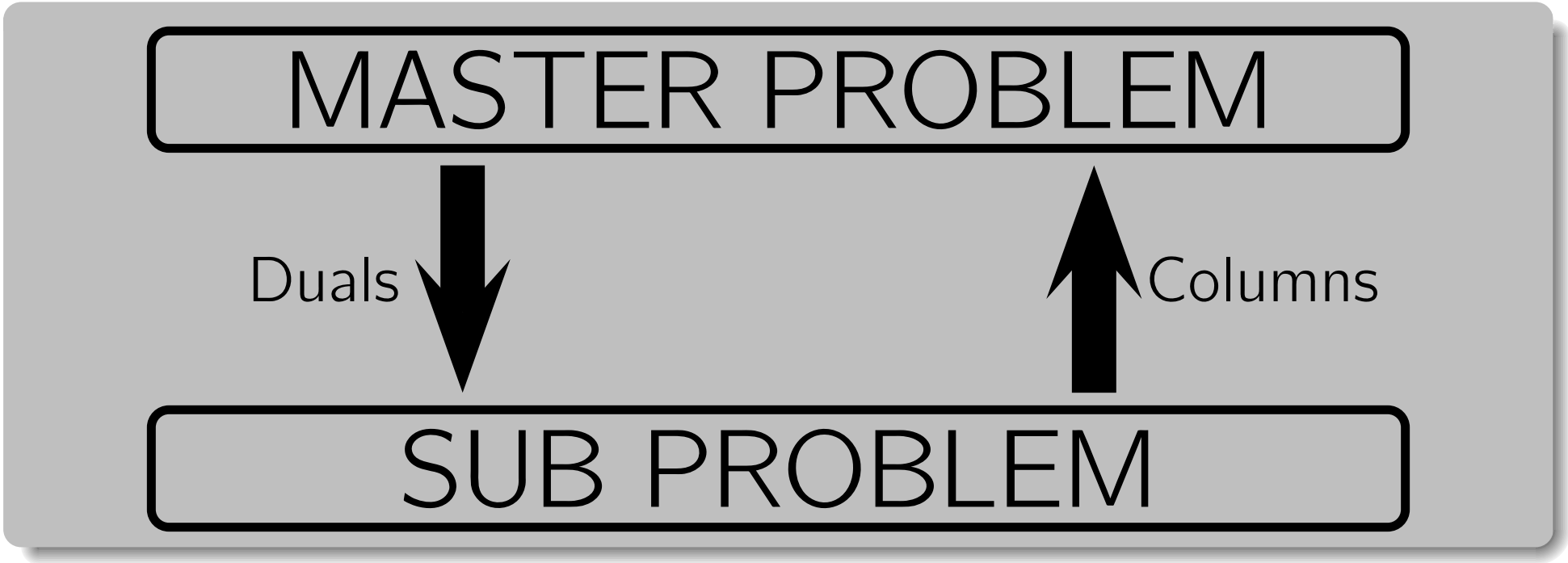
[1]Department of Management Engineering
Technical University of Denmark

42134 Advanced Topics in Operations Research



DTU Management Engineering
Department of Management Engineering

Column Generation



Todays topics

- Branching strategies
- Efficient solution of the master problem

Branching Strategies

Why do we need to branch?

Branching is necessary because we relax our Integer Linear Program in order to solve it efficiently:

- Fractional solutions (lp relaxation)
- Other infeasibilities (combinatorial relaxation)

Branching stategies

- Variable branching
- Constraint branching
- Follow-on branching/"subproblem" branching

Conventional Variable Branching

Strategy

- Select x_j where $0 < x_j < 1$.
  - 1-branch: set $x_j = 1$. Strong branch.
  - 0-branch: set $x_j = 0$. Weak branch.
    - ★ Objective usually unaffected.
    - ★ Subproblem not trivial to solve.
- The bounding process is ineffective.

Constraint Branching I

Definition

- Branch using the Ryan and Foster constraint branch
- Suppose constraints p and r are covered together at fractional value in the lp optimum.
  - At least one pair, p and r will exist in a fractional solution.
- Let $J(p, r) = \{j : a_{pj} = 1 \text{ and } a_{rj} = 1, j = 1, 2, \ldots, n\}$
  - Then in an integer solution *either* p and r must be covered together, *or* p and r cannot be covered together.

Constraint Branching II

Strategy

- Find constraints p and r with $0 < \sum_{j \in J(p,r)} x_j < 1$.
- Often you will try to maximize the sum in order to get close to an integer solution.
- In the 1-branch force p and r to be covered together by setting $x_j = 0$ for all columns j only covered by only one of the constraints.
- In the other branch set $x_j = 0$ for columns j in $J(p, r)$.
- The solution of the subproblem then needs to enforce the decisions made.

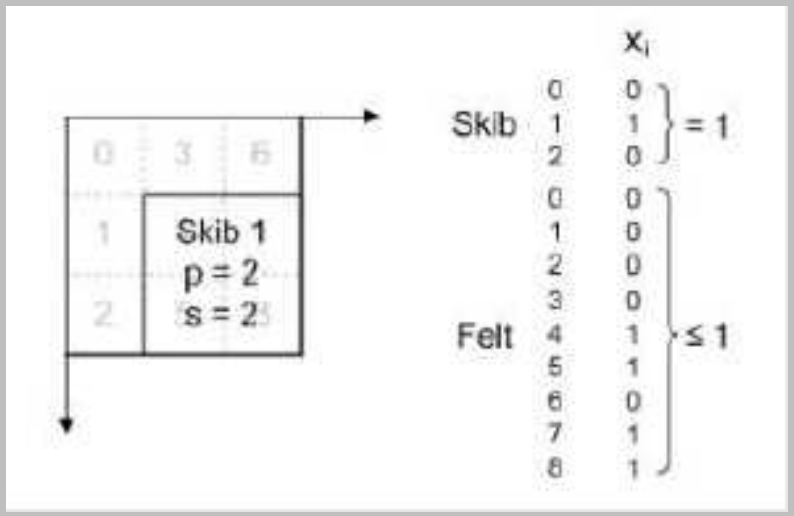
Examples

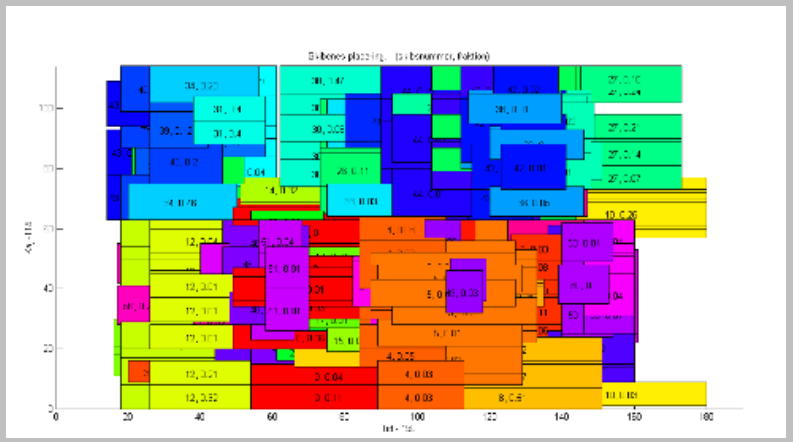
Applications

- Vehicle routing
- Berth Scheduling problem
- Assignment problem with GUB constraints

Berth Scheduling Problem

Constraint branching

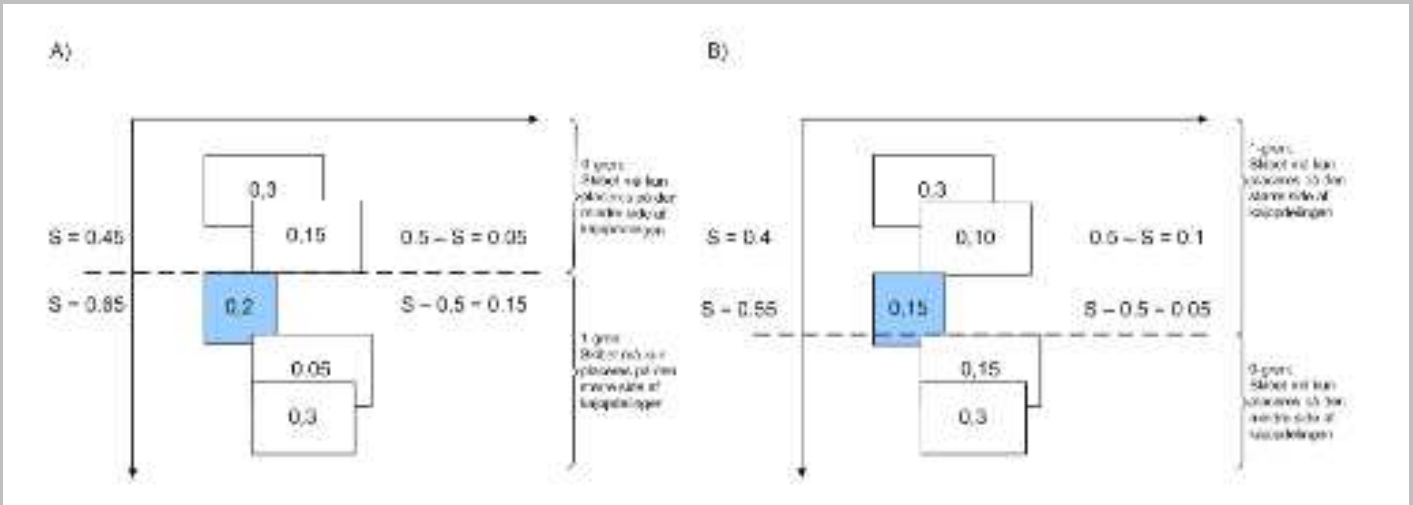




- In the 1-branch the ship is forced to occupy this location in time
- In the 0-branch the ship is forced away from this location in time

Berth Scheduling Problem - Revisited

Alternative branching strategy



Follow-on branching

Strategy

- As an alternative approach branching strategies can work on the subproblem
- Follow-on branching got its name from its application in routing and scheduling applications
- Given a fractional solution. There will be at least one arc (i, j) in the graph of the subproblem with a fractional flow.
  - In the 1-branch a path entering i will be forced to continue to j.
  - In the 0-branch a path entering i can continue to any other node except j.
- Follow-on branching is not "symmetric".
- On the other hand it can be implemented very efficiently by making changes in the subproblem.

Dynamic Constraint Aggregation
Master Problem Characteristics

- Large GSPP of these consists of more than a billion variables and more than a thousand constraints (almost all of them being set partitioning constraints).
- Such a large number of set partitioning constraints and the presence of columns having more than 10 nonzero elements usually yield **high degeneracy** in the restricted master problem.
- This will slow down the column generation process.
- The simplex algorithm that solves the restricted master problem will experience a high percentage of degenerate variables in the basic feasible solution and it will execute many degenerate pivots.

Using The Structure

Motivation

- An idea is to reduce the number of constraints in the restricted master problem.
- This will make the restricted master problem much easier to solve.
- Partly because the number of degenerate pivots are reduced.

Underlying assumption

- In crew scheduling it is observed that crews do not change their vehicles very often.
- Re-optimized solutions deviate slightly from planned ones.
- Consequently many consecutive tasks will remain grouped.

Notation

- A set partitioning constraint is associated with a task.
- A task is accomplished by a commodity.
- The main variables are associated with paths that are feasible with regard to a set of predefined rules.
- A path contains an ordered sequence of tasks and possibly other activities.

Examples

- In crew pairing a task would be a flight, a commodity would be a crew member and a path is a legal pairing.
- In vehicle routing a task is a visit to a customer, a commodity is a vehicle and a path is a route.

Generic Formulation

The Generic Master Problem

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} c_p^k \theta_p^k + M \sum_{w \in W} Y_w$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} a_{wp}^k \theta_p^k + Y_w = 1 \quad \forall w \in W$$

$$\theta_p^k \geq 0 \quad \forall p \in P^k, k \in K$$

$$Y_w \geq 0 \quad \forall w \in W$$

Notice

- Y_m is an artificial variable that guarantee problem feasibility.
- The MP is feasible and bounded.

The Basic Concepts

Equivalence

Given a set of paths C, two tasks w_1 and w_2 are equivalent with respect to C if every path in C covers both w_1 and w_2, or none of them.

Equivalence classes

- This relations partitions tasks into equivalence classes.
- Let L be the set of classes, W_l the subset of tasks in class $l \in L$, and $Q = \{W_l : l \in L\}$.

Compatibility

Let us define compatibility criteria between partition Q and the θ_p^k path variables. Let p be a path in $P^k, k \in K$ and T_p the set of task covered by the path.

- Path p is said to be compatible with the equivalence class $l \in L$ if $W_l \cap T_p$ is either the empty set or equal to W_l.
- Path p is compatible with partition Q if it is compatible with all the equivalence classes in L.
- If p is compatible with the partition Q we say that θ_p^k or its corresponding column is compatible with Q.

Basic Idea

- Instead of using the traditional restricted master problem with all the constraints this approach relies on a so-called aggregated restricted master problem (ARMP).
- ARMP considers smaller subsets of variables and constraints than RMP.

Representative task

- For each equivalence class W_l the ARMP only contains one constraint.
- The task associated with this constraint is denoted the representative task.

Aggregated RMP

- As a consequence we are restricted in adding columns to the ARMP. We can only add columns that are compatible with the partition Q.
- The task aggregation needs to be adjusted dynamically throughout the solution process because we do not know a priori which tasks will be consecutive in the optimal paths.
- We allow for dynamically to update Q and consequently constructing an new ARMP. We say that ARMP is restricted to partition Q and denote it ARMP_Q.

Remaining issues

Initial solution

We initially need some paths in the set C. These can be generated by a heuristic or taken from a planned solution.

Dual variables

- As a result of solving the ARMP_Q we get aggregated dual variables $\hat{\alpha}_l$ for all representative tasks w_l. But in the subproblem I need dual variables for each of the original tasks.
- To do so, the following linear system needs to be solved.

$$\sum_{w \in W_l} \alpha_w = \hat{\alpha}_l \ \forall l \in L$$

Algorithm Description

Main Program

while *True* **do**
  repeat
    $(x, \hat{\alpha}, Z) \leftarrow$ solve ARMP_Q
    compute duals α from $\hat{\alpha}$
    $P'' \leftarrow \text{oracle}(\alpha)$
    if $P'' = \emptyset$ **then**
      STOP
    $P' \leftarrow P' \cup P''$
  until $P''_Q = \emptyset$ *or modify*(α)
  repartition Q

repartition Q

I is a nonempty set of negative reduced cost columns incompatible with Q
if $Z = Z_{old}$ *or* $\exists I : Y_{w_I} > 0$ **then**
  $C \leftarrow C \cup I$
else
  $C \leftarrow B \cup I$
redefine Q according to C
$Z_{\text{old}} \leftarrow Z$

Redefinition of Q

repartition Q

I is a nonempty set of negative reduced cost columns incompatible with Q
if $Z = Z_{old}$ *or* $\exists I : Y_{w_I} > 0$
then
  $C \leftarrow C \cup I$
else
  $C \leftarrow B \cup I$
redefine Q according to C
$Z_{\text{old}} \leftarrow Z$

Two alternatives

- **Alternative 1**: Invoked if last partition did not improve the objective function or the optimal solution is infeasible.
  - Expand based on the previous set of paths.
- **Alternative 2**: Invoked if successful in decreasing the objective function value.
  - Expand based on the non-degenerate basic columns.

Partition Handling

- After every minor iteration the algorithm must decide if the partition must be redefined.
- Beside updating when P''_Q is empty a strategy is implemented to redefine Q when it seems profitable.
- This leaves room for developing your own heuristic. The authors suggest:

$$\text{modify}(\alpha) = \begin{cases} \text{true} & \text{if } \{p \in \bar{P}'_Q : \bar{c}_p(\alpha) < 0\} \neq \emptyset \text{ and} \\ & r = \dfrac{\min_{p \in P''_Q} \bar{c}_p(\alpha)}{\min_{p \in \bar{P}'_Q} \bar{c}_p(\alpha)} < \lambda \\ \text{false} & \text{otherwise} \end{cases}$$

- Problem specific knowledge could be included here.

Dual Variables Disaggregation

Strategy

- One difficulty with the dynamic constraint aggregation method is that it does not provide a complete dual solution.
- We need a complete dual solution for solving the pricing problem.
- Instead it provides an aggregated dual solution $\hat{\alpha}$.
- To find a disaggregated one we need a feasible solution α to

$$\sum_{w \in W_l} \alpha_w = \hat{\alpha}_l \ \forall l \in L \tag{1}$$

Finding a disaggregated dual

Simple solution

Setting $\alpha_w = \hat{\alpha}_l / \|W_l\| \; \forall w \in W_l, l \in L$ defines a feasible solution to this system. It is although a crude and inefficient solution.

Complex solution

- For every generated incompatible column p it must hold that

$$\sum_{w \in W} a_{wp} \alpha_w \leq c_p$$

- These can be added to (1).
- Only problem is that this problem is potentially as hard to solve as our original master problem.
- An intelligent restriction of constraints added to (1) makes it possible to solve the problem as a shortest path problem.

Computational Experimentation

Test setup

- Computational experiments are conducted on instances of the vehicle and crew scheduling problem in urban mass transit systems.
- Problem consists of determining bus and crew schedules simultaneously for a given time table.
- Objective function is to minimize total cost.
- Tests limited to solving the linear relaxation of the instances.
- As drivers can only change buses after a break.
- Buses must be assigned to trips and drivers to segments.

Instance

Instances

- Instances were randomly generated.
- In total 32 instances containing between 20 to 160 trips are generated.
- The number of segments per trip is 2, 4, 6 or 8.
- The number of task in an instance is the number of trips times the number of segments.

Results

- Reduction factor in solution time is between 1.7 and 12.2.
- The factors grows with the size of the problems.
- For problems with more than 700 tasks the reduction factor is at least 3, and at least 4 for problems with more than 1000 tasks.
- In the standard column generation method more than 70% of the time is used solving the master strongly motivating dynamically aggregating some MP constraints.
- Number of constraints are reduced by an average of 39% and the MP time by up to 90%.