Solving Super-Size Problems with Optimization

Horia Tipi
Principal Consultant
FICO
horiatipi@fico.com

Very Large Scale Optimization Problems

» Schedule crews for 3,400 daily flights in 40 countries

» Buy ads in 10-15 local publications across 40,000 zip codes

» Pick one of 742 trillion choices in creating the NFL schedule

» Select 5 offers out of 1,000 for each of 25,000,000 customers

» Place 1,000s of SKUs on dozens of shelves in ~2,000 stores

» Decide among 200,000,000 maintenance routing options

» Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Must solve (close to) optimally, or your competition will

Benefits

- Regulatory and labor compliance while maximizing productivity
- Increased decision precision, saving millions of dollars per year
- Create schedules that satisfy all stakeholders
- Improve response rates from 1-2% to 5-10%
- Consistent and faster retail planning results in hundreds of thousands of dollars in both cost reduction and revenue increase per store
- Improve rental fleet utilization to save $19M
- Food manufacturing efficiency improvements of millions of dollars per production line

Agenda

Large-Scale Problem Types and Solution Approaches

Modeling Tool Requirements

Large Scale Mixed Integer Programming Problems

»Schedule crews for 3,400 daily flights in 40 countries
»Buy ads in 10-15 local publications across 40,000 zip codes
»Pick one of 742 trillion choices in creating the NFL schedule
»Select 5 offers out of 1,000 for each of 25,000,000 customers
»Place 1,000s of SKUs on dozens of shelves in ~2000 stores
»Decide among 200,000,000 maintenance routing options
»Plan weekly production levels for several years ahead

Q: Why are these problems difficult?

A: Because requirements are complex:

» Combination of continuous, binary, and integer decision variables
» Often multi-objective (weighted objectives or goal programming)
» Often with soft constraints (nice to have)
» Often infeasible unless most soft constraints are relaxed
» Lots of unknowns

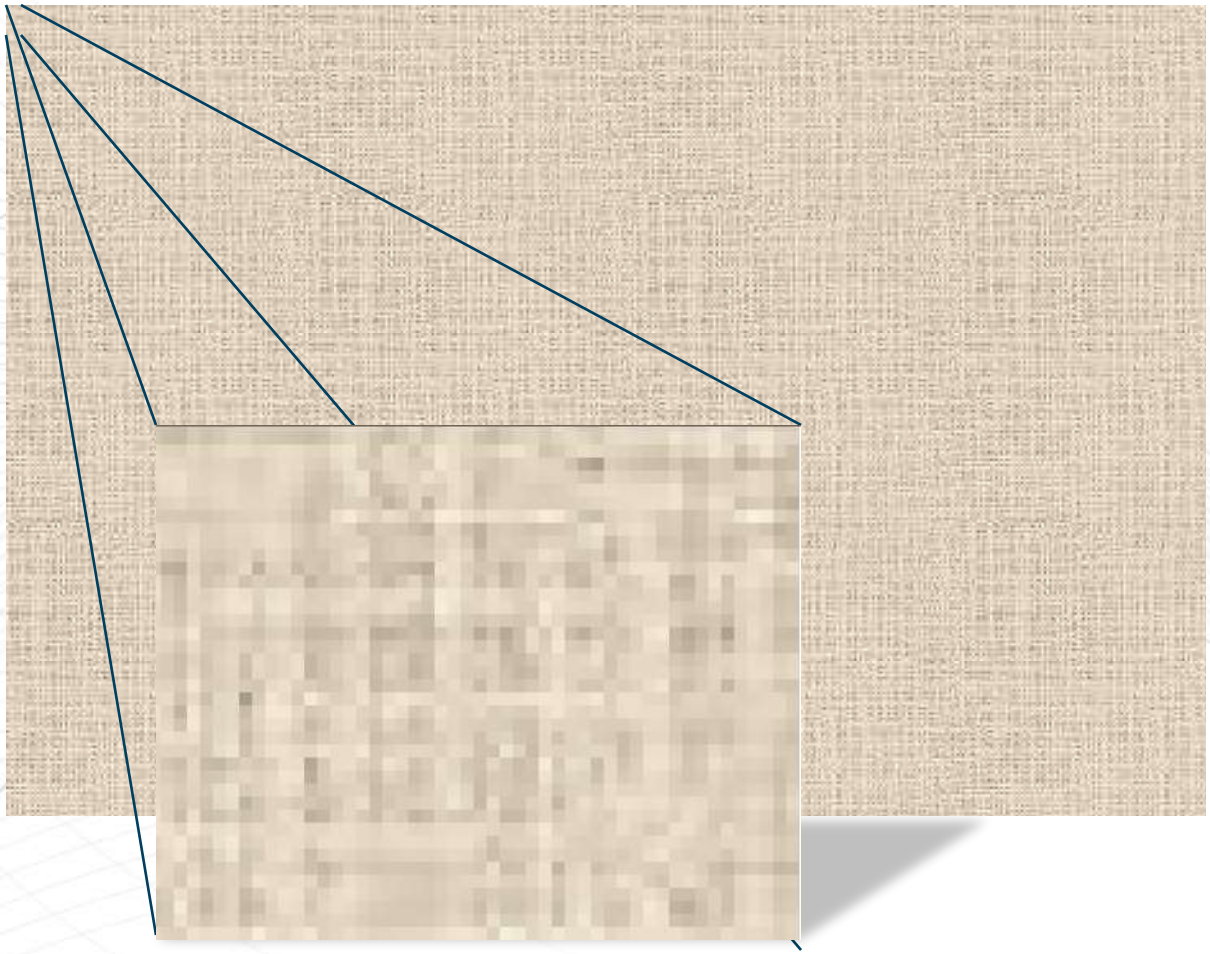
Optimization Problem Structure

Decision variables (up to 100s of millions)

a.k.a. Columns

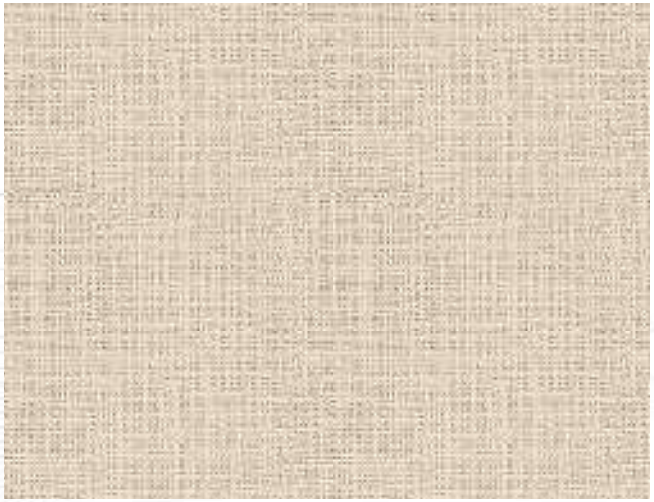
Constraints (up to 10s of millions)

a.k.a. Rows



Types of large MIP problems: No discernible shape

Problems with tightly coupled rows and columns



Signature:	Every decision almost directly impacts every other decision
Examples:	Multi-party Exchanges, Scheduling, TSP
The bad news:	Can be hard to solve, even for medium-size problems
The good news:	Easy to model; Advanced solver tuning can pay off
Your best bet:	Get the best solver and tune it; Identify patterns and reformulate through substitution/decomposition if possible

Very Large Scale Optimization Problems

» Schedule crews for 3,400 daily flights in 40 countries

» Buy ads in 10-15 local publications across 40,000 zip codes

» Pick one of 742 trillion choices in creating the NFL schedule

» Select 5 offers out of 1,000 for each of 25,000,000 customers

» Place 1,000s of SKUs on dozens of shelves in ~2,000 stores

» Decide among 200,000,000 maintenance routing options

» Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Types of large MIP problems: Columns >> Rows

An extremely large number of columns

Signature:	More variables than any computer's memory could fit
Examples:	Crew pairing, sports scheduling, vehicle routing
The bad news:	Implementations may have to sacrifice optimality
The good news:	Parallel implementations work well in practice
Your best bet:	Column generation (master problem and iterative sub-problems)

Very Large Scale Optimization Problems

» Schedule crews for 3,400 daily flights in 40 countries

» Buy ads in 10-15 local publications across 40,000 zip codes

» Pick one of 742 trillion choices in creating the NFL schedule

» Select 5 offers out of 1,000 for each of 25,000,000 customers

» Place 1,000s of SKUs on dozens of shelves in ~2,000 stores

» Decide among 200,000,000 maintenance routing options

» Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Column generation in Xpress-Mosel

```
while(true) do
    minimize(XPRS_LIN, MinRolls)        ! Solve the LP
    savebasis(bas)                      ! Save the current basis
    objval:= getobjval                  ! Get the objective value
                                        ! Get the solution values
    forall(j in 1..npatt)  soluse(j):=getsol(use(j))
    forall(i in WIDTHS)  dualdem(i):=getdual(Dem(i))
                                        ! Solve a knapsack problem
    zbest:= knapsack(dualdem, WIDTH, MAXWIDTH, xbest) - 1.0
    write("Pass ", npass, ": ")
    if zbest = 0 then                   ! Same as zbest <= EPS
      writeln("no profitable column found.\n")
      break
    else
      show_new_pat(zbest, xbest)        ! Print the new pattern
      npatt+=1
      create(use(npatt))                ! Create a new var. for this pattern
      use(npatt) is_integer
      MinRolls+= use(npatt)             ! Add new var. to the objective
      dw:=0
      forall(i in WIDTHS)
        if xbest(i) > 0 then
         Dem(i)+= xbest(i)*use(npatt)   ! Add new var. to demand constr.s
         dw:= maxlist(dw, ceil(DEMAND(i)/xbest(i) ))
        end-if
      use(npatt) <= dw                  ! Set upper bound on the new var.
      loadprob(MinRolls)                ! Reload the problem
      loadbasis(bas)                    ! Load the saved basis
    end-if
    npass+=1
  end-do
```

Column generation in Xpress-Mosel

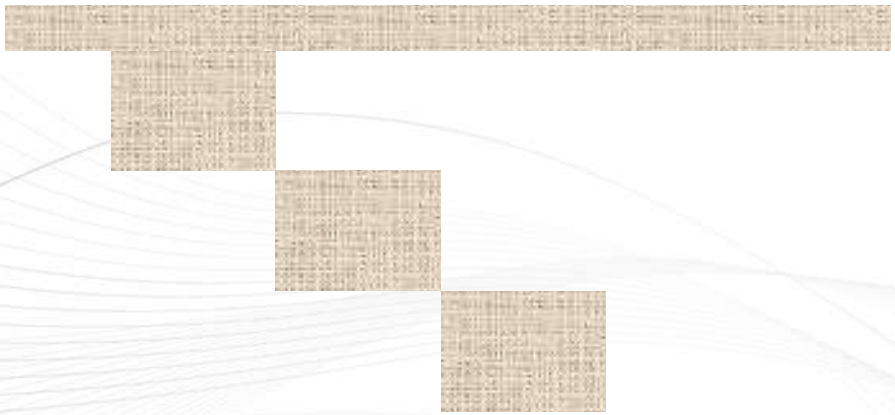
```
!*********************************************************************
!  Solve the integer knapsack problem
!    z = max{cx : ax<=b, x in Z^N}
!  with b=MAXWIDTH, N=NWIDTHS
!
! We reset the knapsack constraints to 0 at the end of this function
! so that they do not unnecessarily increase the size of the main
! problem. The same is true in the other sense: hiding the demand
! constraints while solving the knapsack problem makes life easier
! for the optimizer, but is not essential for getting the correct
! solution.
!*********************************************************************
 function knapsack(C:array(range) of real,
                   A:array(range) of real,
                   B:real,
                   xbest:array(range) of integer):real
  declarations
   x: array(WIDTHS) of mpvar            ! Knapsack variables
  end-declarations
  with mpproblem do                     ! Create a local subproblem
 ! Define the knapsack problem
   forall(j in WIDTHS) x(j) is_integer
   sum(j in WIDTHS) A(j)*x(j) <= B
   maximize(sum(j in WIDTHS) C(j)*x(j))
   returned:=getobjval
   forall(j in WIDTHS) xbest(j):=round(getsol(x(j)))

  end-do

 end-function
```

Types of large MIP problems: Master/Subproblems

Problems consisting of independent sub-problems connected by global constraints



Signature:	Different independent business units with shared resources
Examples:	Multi-period production planning, hierarchical packing
The bad news:	Difficulty depends on the number of global constraints
The good news:	Parallel implementations work well in practice
Your best bet:	Dantzig-Wolfe decomposition, Hybrid algorithms, heuristics

Dantzig-Wolfe in Xpress-Mosel

```
!-------------------------------------------------------------
! Process the proposal generated by a subproblem
 procedure process_sub_result
  declarations
   f: integer                          ! Factory index
                                       ! Solution values of the proposal:
   sol_make: array(PRODS,TIME) of real      ! Amount of products made
   sol_sell: array(PRODS,TIME) of real      ! Amount of product sold
   sol_buy: array(RAW,TIME) of real         ! Amount of raw mat. bought
   sol_pstock: array(PRODS,1..NT+1) of real ! Product stock levels
   sol_rstock: array(RAW,1..NT+1) of real   ! Raw mat. stock levels
   pc: real                            ! Cost of the proposal
  end-declarations
 ! Read proposal data from memory
  initializations from "mempipe:noindex,sol"
   f
   sol_make sol_sell sol_buy sol_pstock sol_rstock
   pc
  end-initializations
 ! Add the new proposal to the master problem
  nPROP(f)+=1
  create(weight(f,nPROP(f)))
  forall(p in PRODS,t in TIME) do
   Prop_make(p,f,t,nPROP(f)):= sol_make(p,t)
   Prop_sell(p,f,t,nPROP(f)):= sol_sell(p,t)
  end-do
  forall(r in RAW,t in TIME) Prop_buy(r,f,t,nPROP(f)):= sol_buy(r,t)
  forall(p in PRODS,t in 1..NT+1) Prop_pstock(p,f,t,nPROP(f)):= sol_pstock(p,t)
  forall(r in RAW,t in 1..NT+1) Prop_rstock(r,f,t,nPROP(f)):= sol_rstock(r,t)
  Prop_cost(f,nPROP(f)):= pc
  writeln("Sol. for factory ", f, ":\n  make:   ", sol_make, "\n  sell:   ",
          sol_sell, "\n  buy:    ", sol_buy, "\n  pstock: ", sol_pstock,
       "\n  rstock: ", sol_rstock)
 end-procedure
```

Dantzig-Wolfe in Xpress-Mosel

```
!-------------------------------------------------------------
! (Re)solve the master problem
 procedure solve_master(phase: integer)
  forall(f in FACT)
   Convex(f):= sum (k in 1..nPROP(f)) weight(f,k) = 1
  if phase=1 then
   forall(p in PRODS,t in TIME)
    MxSell(p,t):=
     sum(f in FACT,k in 1..nPROP(f)) Prop_sell(p,f,t,k)*weight(f,k) -
      excessS <= MXSELL(p,t)
   minimize(excessS)
  else
   forall(p in PRODS,t in TIME)
    MxSell(p,t):=
     sum(f in FACT,k in 1..nPROP(f)) Prop_sell(p,f,t,k)*weight(f,k) <=
      MXSELL(p,t)
   maximize(sum(f in FACT, k in 1..nPROP(f)) Prop_cost(f,k) * weight(f,k))
  end-if
  writeln("Master problem objective: ", getobjval)
  write("  Weights:")
  forall(f in FACT,k in 1..nPROP(f)) write(" ", getsol(weight(f,k)))
  writeln
 end-procedure
```

Dantzig-Wolfe in Xpress-Mosel

```
! (Re)solve this model until it is stopped by event "PHASE_3"
 repeat
  wait
  ev:= getnextevent
  Phase:= getclass(ev)
  if Phase=PHASE_3 then                ! Stop the execution of this model
   break
  end-if
  Price_convex:= getvalue(ev)          ! Get new pricing data
  if Phase<>PHASE_0 then
   initializations from "raw:noindex"
    Price_sell as "shmem:Price_sell"
   end-initializations
  end-if

 ! (Re)solve this model
  if Phase=PHASE_0 then
   maximize(Profit)
  elif Phase=PHASE_1 then
   maximize(sum(p in PRODS,t in TIME) Price_sell(p,t)*sell(p,t) + Price_convex)
  else         ! PHASE 2
   maximize(
    Profit - sum(p in PRODS,t in TIME) Price_sell(p,t)*sell(p,t) -
     Price_convex)
  end-if
  writeln("Factory ", Factory, " - Obj: ", getobjval,
          " Profit: ", getsol(Profit), " Price_sell: ",
          getsol(sum(p in PRODS,t in TIME) Price_sell(p,t)*sell(p,t) ),
          " Price_convex: ", Price_convex)
  fflush

  if getobjval > TOL then              ! Solution found: send values to master
   process_solution
  elif getobjval <= TOL then           ! Problem is infeasible (Phase 0/1) or
   send(EVENT_FAILED,0)                ! no improved solution found (Phase 2)
  else
   send(EVENT_READY,0)
  end-if
 until false
```

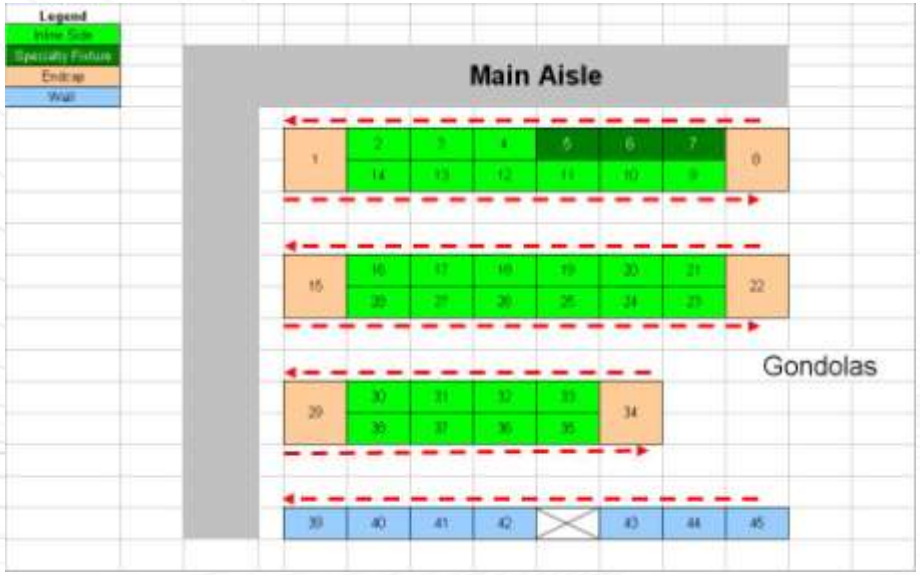
Very Large Scale Optimization Problems

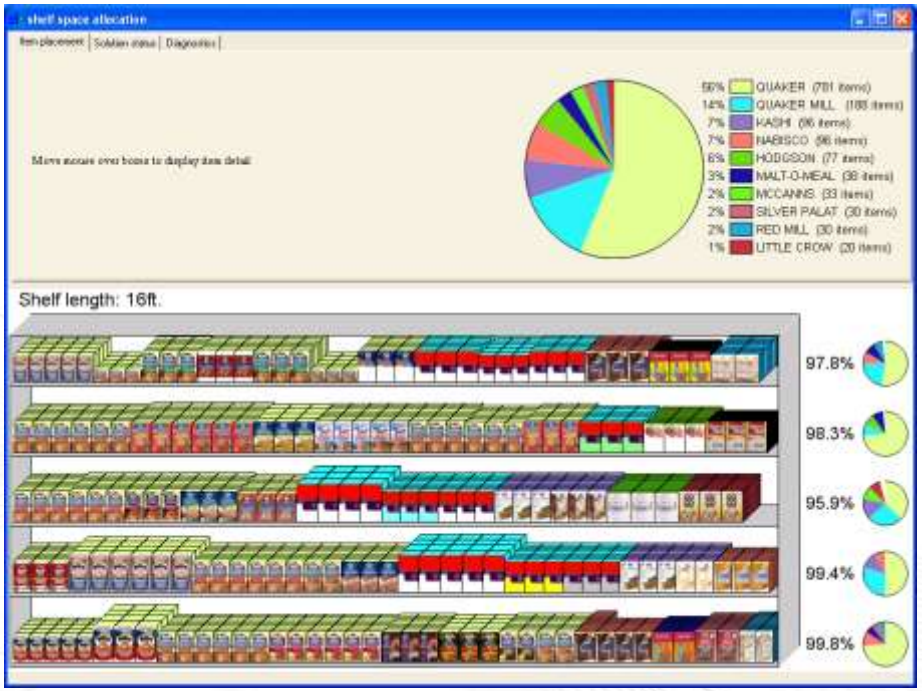
- Schedule crews for 3,400 daily flights in 40 countries
- Buy ads in 10-15 local publications across 40,000 zip codes
- Pick one of 742 trillion choices in creating the NFL schedule
- Select 5 offers out of 1,000 for each of 25,000,000 customers
- Place 1,000s of SKUs on dozens of shelves in ~2,000 stores
- Decide among 200,000,000 maintenance routing options
- Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Macro and Micro planning





»The problem naturally decomposes into a macro problem (overall layout) and many micro problems (individual shelf layout)

Very Large Scale Optimization Problems

» Schedule crews for 3,400 daily flights in 40 countries

» Buy ads in 10-15 local publications across 40,000 zip codes

» Pick one of 742 trillion choices in creating the NFL schedule

» Select 5 offers out of 1,000 for each of 25,000,000 customers

» Place 1,000s of SKUs on dozens of shelves in ~2,000 stores

» Decide among 200,000,000 maintenance routing options

» Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Production planning at a US food company

Given

- Demand and
- Available capacity (e.g. number of production lines and corresponding capacities)

Provide

- *optimized* production schedule at the product level for a given period of time

Objectives

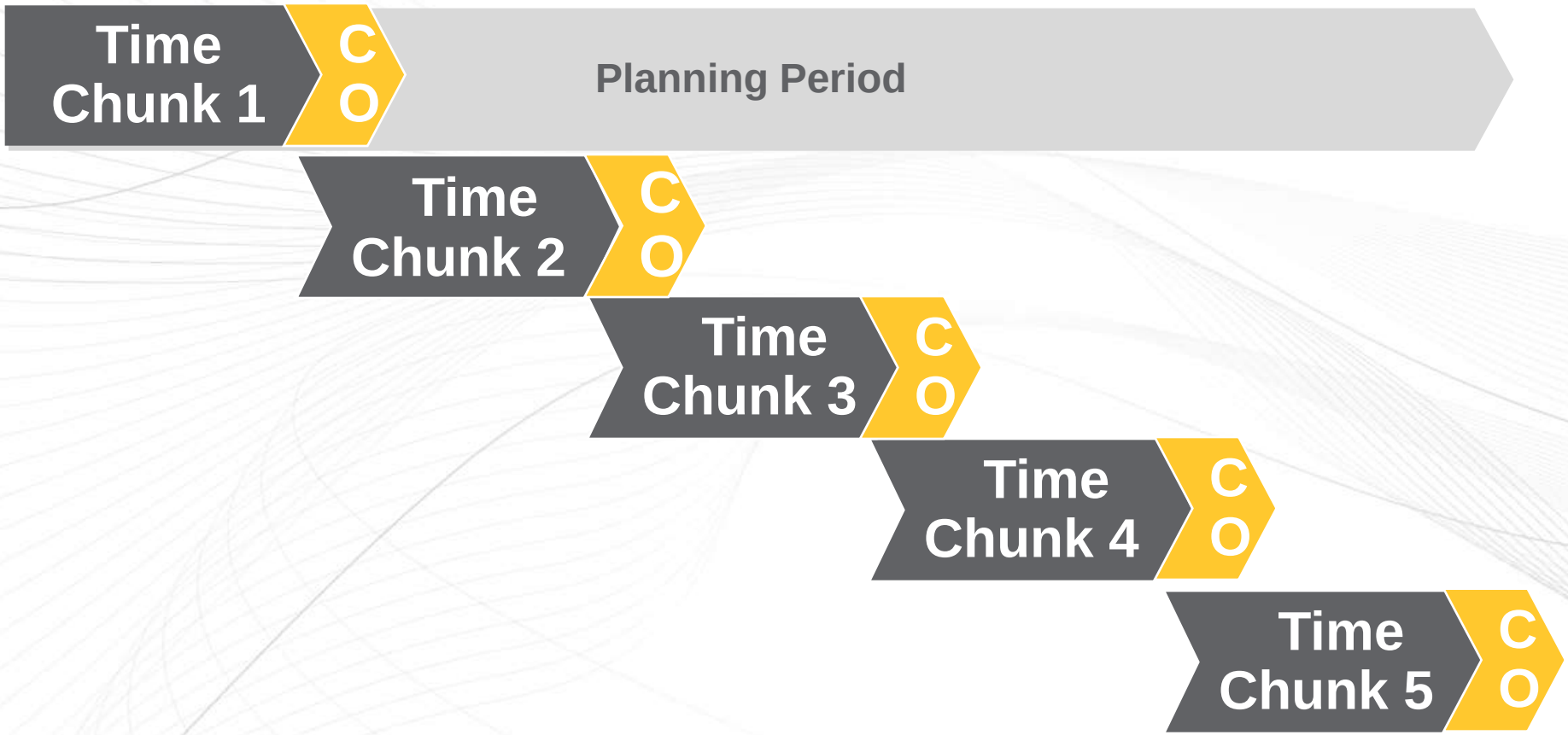
- Meet demand as closely as possible
- Minimize setup cost of production lines when switching between two different products
- Minimize cost of labor
- Minimize inventory
- Minimize staleness

Chronological Decomposition Heuristic

» The planning period is split up into chunks

» Cross-over intervals are added

» Each time chunk is solved separately

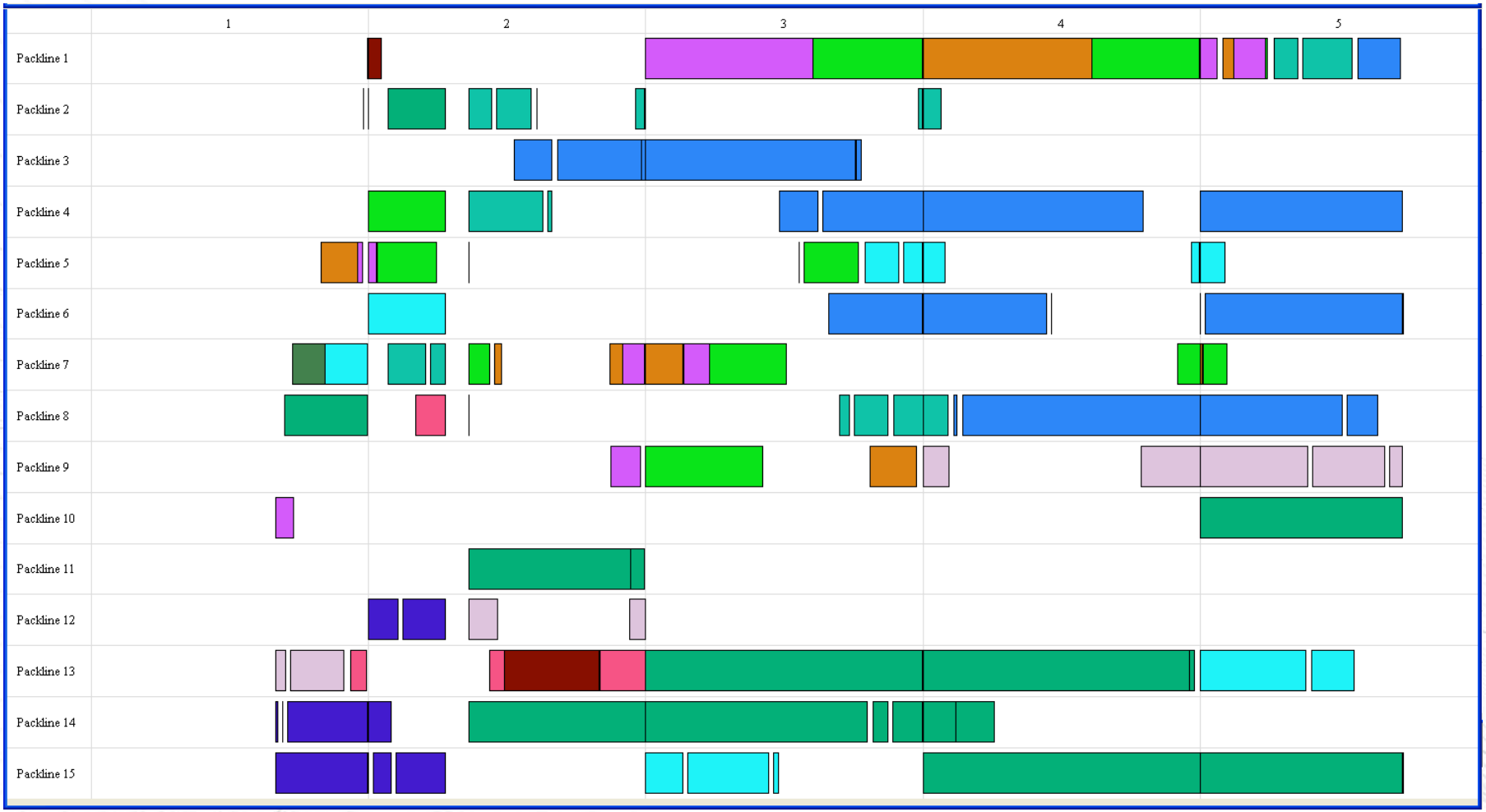


Combining partial solutions

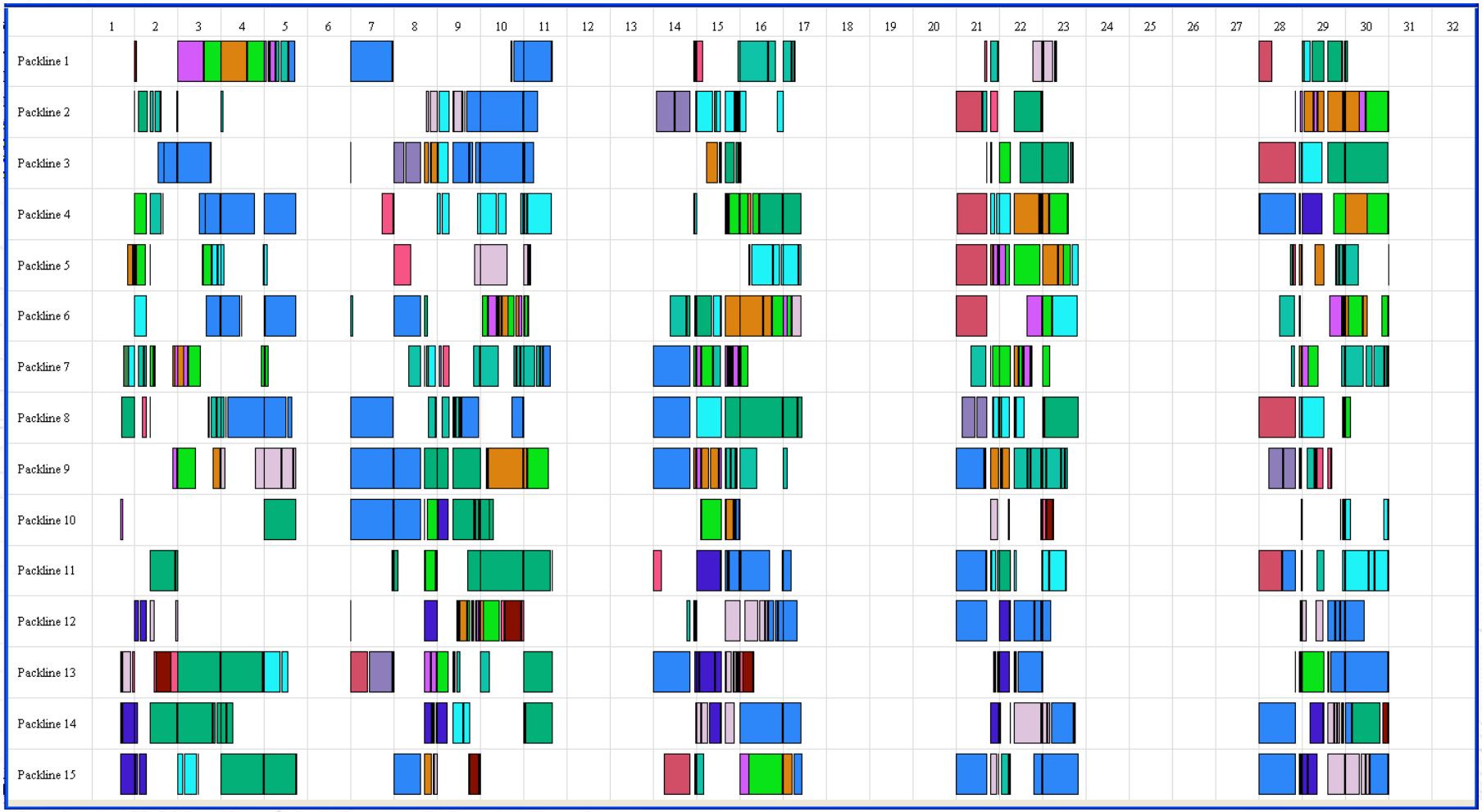
- … and the solutions are glued together
- If several solutions for each time chunk are available these solutions can be combined to different solutions for the complete problem
- A heuristic tries to improve the solutions locally

see J.D. Kelly, Chronological Decomposition Heuristic for Scheduling: A Divide & Conquer Method

A one week plan

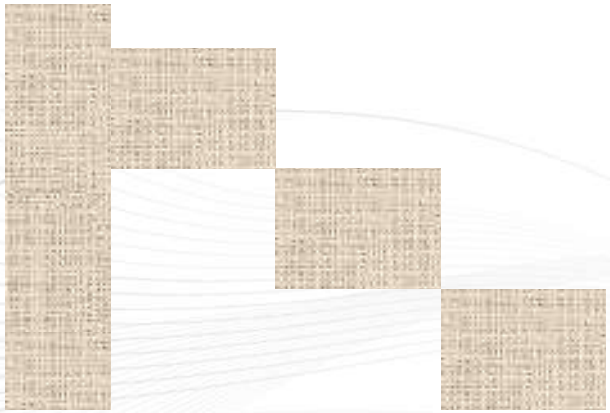


A five week plan



Types of large MIP problems: Multi-stage stochastic

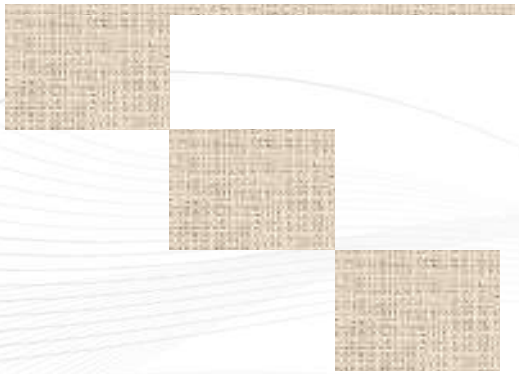
Problems consisting of independent sub-problems connected by common columns



Signature:	Stage 1 decisions in multi-period discrete stochastic problems
Examples:	Problems with diverging future expectation (energy, finance)
The bad news:	Combinatorial explosion when large number of stages
The good news:	Decisions can be re-optimized when Stage 2 becomes Stage 1
Your best bet:	Benders decomposition, simulation

Types of large MIP problems: Large assignment

Assignment problems applied to very large quasi-homogeneous populations, with continuous global constraints



Signature:	Billions/Trillions of simple assignment decisions
Examples:	Marketing problems, Credit line assignment, Collections
The bad news:	May have to trade some optimality for speed
The good news:	Can decompose aggressively
Your best bet:	Random partitioning, but keep an eye on solution quality; use Dantzig-Wolfe or alternative if optimality is crucial

Very Large Scale Optimization Problems

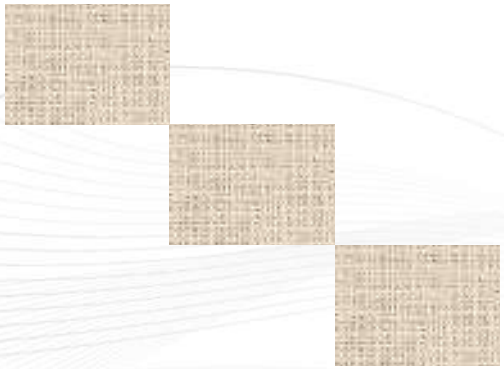
- Schedule crews for 3,400 daily flights in 40 countries
- Buy ads in 10-15 local publications across 40,000 zip codes
- Pick one of 742 trillion choices in creating the NFL schedule
- Select 5 offers out of 1,000 for each of 25,000,000 customers
- Place 1,000s of SKUs on dozens of shelves in ~2,000 stores
- Decide among 200,000,000 maintenance routing options
- Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Types of large MIP problems: Aggregated

Independent problems – when decentralized, individual planning makes sense



Signature:	Heterogeneous production units with independent budgets
Examples:	Routing to specialized repair facilities; production without inventory
The bad news:	Almost none
The good news:	A few lines of model code is all that's required
Your best bet:	Divide and conquer; Parallelize

Very Large Scale Optimization Problems

- Schedule crews for 3,400 daily flights in 40 countries
- Buy ads in 10-15 local publications across 40,000 zip codes
- Pick one of 742 trillion choices in creating the NFL schedule
- Select 5 offers out of 1,000 for each of 25,000,000 customers
- Place 1,000s of SKUs on dozens of shelves in ~2,000 stores
- Decide among 200,000,000 maintenance routing options
- Plan weekly production levels for several years ahead

in pursuit of Conflicting business objectives (goals)

and subject to Multiple conflicting restrictions (constraints)

Summary of Problem Types and Solution Approaches

- Every real-life problem has some exploitable structure
- Known algorithms can be implemented to attack difficult problems
- The MIP solver will do the heavy lifting in finding solutions
- The MIP solver provides proof of optimality or near optimality

Requirements for Very Large Scale Optimization

- Technology enablers:
  - Hardware
    - 64 bit systems
    - Multi-CPU/Multi-core systems
  - Software
    - The fastest LP/MIP solver that:
      - Implements all state of the art cutting, branching and heuristic techniques
      - Allows search customization and fine tuning
      - Provides automatic tuning tools
    - A powerful modeling system that offers:
      - Programming language features (imperative, modular, fast data processing)
      - Built-in decomposition capabilities
      - Built-in parallelism
      - Fast in-memory data transfer
      - Advanced debugging and logging features
      - Rapid visualization for detecting patterns in the problem structure
    - A development environment with:
      - Detailed solver tracking
      - Wizards and a large variety of examples

FICO Xpress Optimization Suite
Key features for large-scale optimization

- Xpress-Optimizer efficiently solves LP and MIP problems with billions of nonzero coefficients
  - Built-in parallelism, advanced cutting, heuristics
- Xpress-Mosel is a high-level modeling language combined with the standard functionality of a programming language
  - Implementation of models and solution algorithms in a single environment; built-in parallelism, model separation, modularity
- Xpress-IVE and Xpress-Tuner support model building
  - Enhanced productivity through the most advanced debugging, profiling, tuning, and execution tracking features on the market

"My team is more productive because they have more access to these kinds of tools. It was a hit in my group when we signed the agreement with FICO."

-- Armando Silva, managing director of operations research, American Airlines

THANK YOU

Q & A

Horia Tipi
Principal Consultant
FICO
horiatipi@fico.com