Experiments with MIP versus CP for Paint Production Scheduling

Bob Hattersley

Opta Consulting Ltd

Copyright © Opta Consulting Ltd 2010

Paint Production Scheduling

- Multiple stages – dispersion, mixing, packing
- Stages coupled – paint needs a vessel
- Disparate equipment in each stage
- Disparate batches to schedule
- Challenging problem

Copyright © Opta Consulting Ltd 2010

Contents

- Description of problem
- Initial MIP model
- CP model and enhancements
- Enhanced MIP model
- And back to CP
- Conclusions

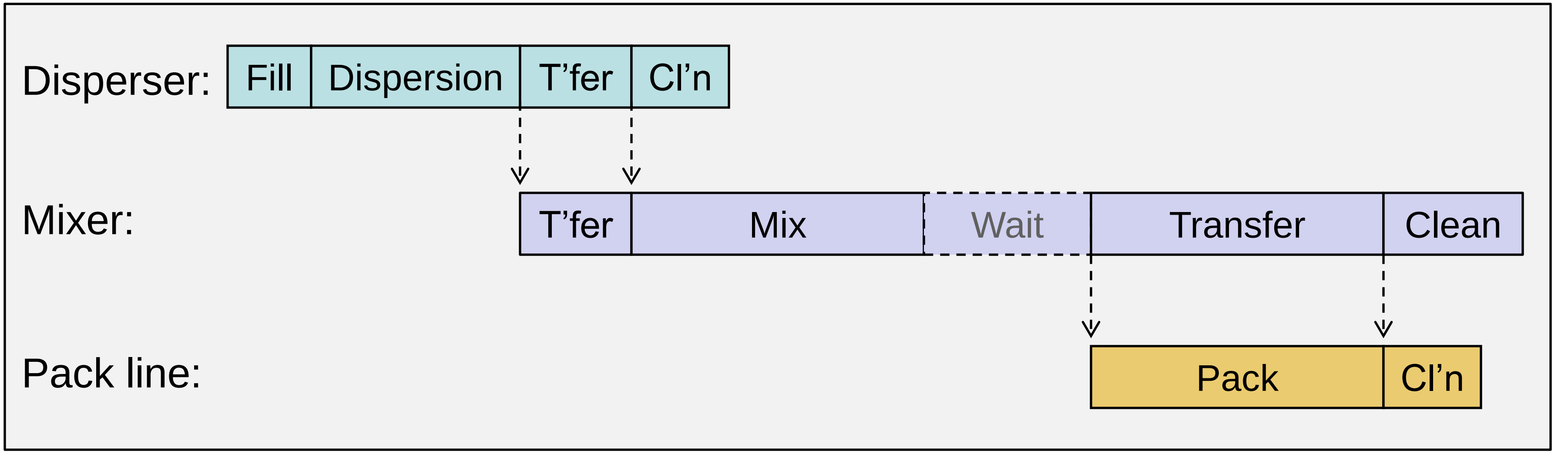
Copyright © Opta Consulting Ltd 2010

The Problem

- 100 batches to schedule (2 weeks)
  - Pre-processed make-to-order + stock
- Due dates
- Choice of (up to) three dispersers
  - Close to 100% utilisation
- Choice of ten mixing vessels
- Choice of (up to) three pack lines
  - Close to 100% utilisation

Copyright © Opta Consulting Ltd 2010

Lifecycle of One Batch



- Durations (except for Wait) are fixed
- Different for each batch
- Different for each facility (except Mix)
- No changeover considerations

Copyright © Opta Consulting Ltd 2010

MIP Model

- Continuous-time model
  - Variables for start times (end times)
- Easy constraints:
  - Selection of facilities for each batch
  - Time offsets
    - Start-end on one facility
    - Start/end from one stage to next
  - Lateness
- Hard: prevent clashes on facilities

Copyright © Opta Consulting Ltd 2010

MIP Model: Prevent Clashes 1

- Disjunctive constraint?
  - Variables: batch a is before b or after b
  - (b after a) $\Rightarrow$ (b starts after a ends)
  - Complicated by facility choice
  - Weak bounds
- Sequencing (vehicle routing)?
  - Variables: batch b follows a on facility
  - Flow constraints
  - Sub-tour elimination
  - Changeover is not an issue – timing is

Copyright © Opta Consulting Ltd 2010

MIP Model: Prevent Clashes 2

- Indexing
  - Variables: batch b runs at index i on facility f
  - Relate batch index start (end) to facility index start (end)
  - Facility index start i+1 ≥ facility index end i
  - Extra start variables by (batch, facility, index)
  - Lower bounds on start by analysis of length (sort shortest first)
  - Then push out pack bounds by offsets
  - Leads to better objective bound and branch decisions

Copyright © Opta Consulting Ltd 2010

Indexing Formulation

b	batch
f	facility
i	index
s	stage

- $index\_run_{bfi} \in \{0,1\}$
  $index\_start_{bfi}$
  $facility\_start_{fi}$
  $batch\_start_{bs}$
- $index\_start_{bfi} \geq Earliest\_start_{fi} \,.\, index\_run_{bfi}$
- $index\_start_{bfi} \leq Latest\_start_{fi} \,.\, index\_run_{bfi}$
- $facility\_start_{fi} = \sum_b index\_start_{bfi}$
- $batch\_start_{bs} = \sum_{f \in s,i} index\_start_{bfi}$
- $facility\_start_{fi} \geq facility\_start_{fi-1}$
  $+ \sum_b Length_{bf} \,.\, index\_run_{bfi}$
  $- \ldots$

Copyright © Opta Consulting Ltd 2010

Problem Size

- All methods lead to quadratic model size
- Impossible to solve in one go
- Solve in steps:
  - From n earliest-due unfixed batches
  - Select m batches, assign facilities and sequence
  - Fix the selected m batches, repeat
- Choice of n, m?

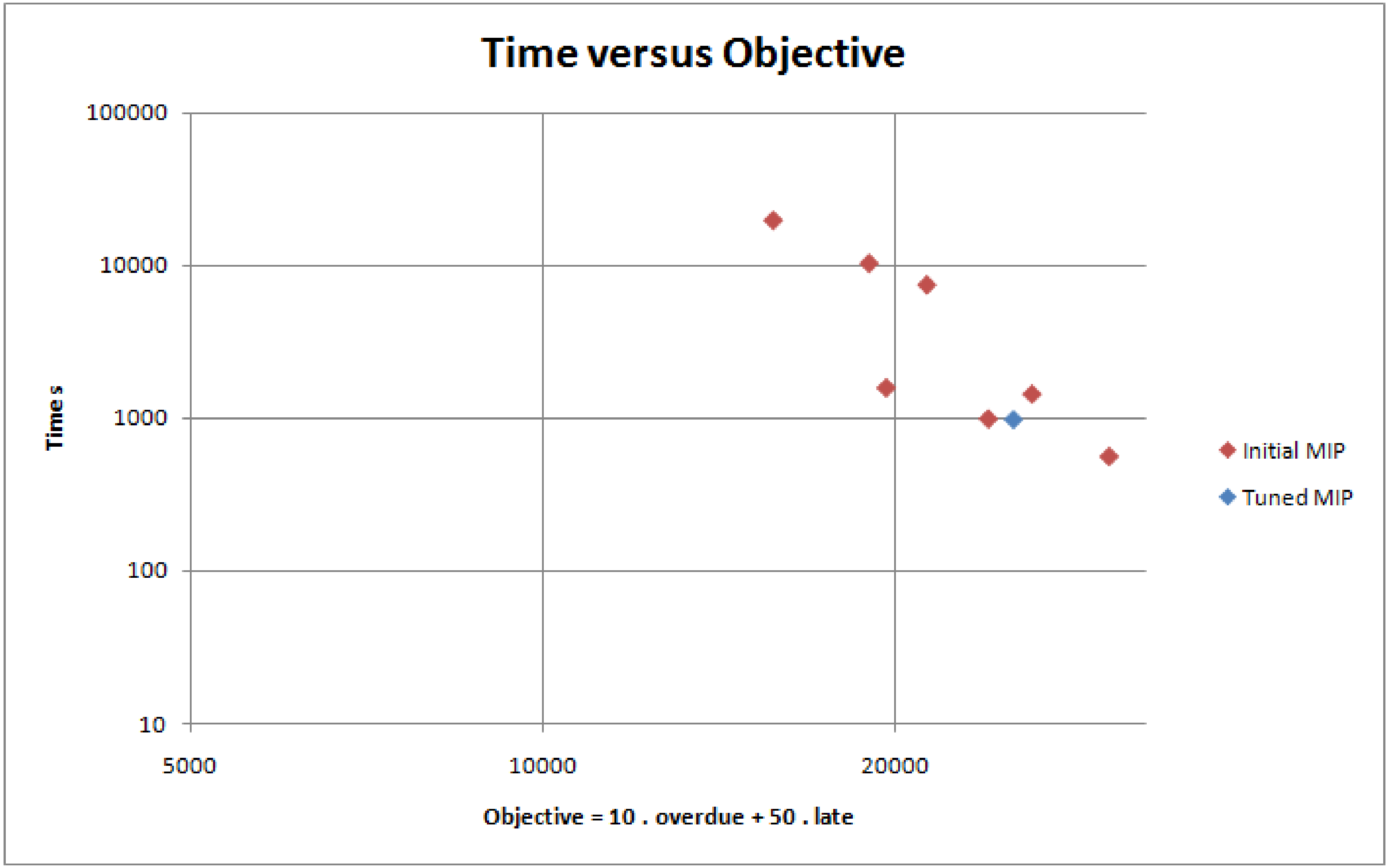
Copyright © Opta Consulting Ltd 2010

Mosel Implementation

- One .mos file
- Define all variables at the start
- In each step:
  - `reset(Problem), with Problem do`
  - Define binaries, fix old choices
  - Constraints including fixed and new batches
  - Solve – Presolve eliminates fixed part
  - Extract choices

Copyright © Opta Consulting Ltd 2010

MIP Results



Copyright © Opta Consulting Ltd 2010

Implementation in Kalis 1

- Variables are simpler and fewer
  - Batch is on facility → Facility that batch is on
  - But cannot use strings as identifiers
  - Have to `setname` explicitly
  - Data rounded to whole hours so all `cpvar`
- Constraints mostly easy to write
  - `equiv(vBatRun(bn)>=1, vBatFac(bn,s)>=1)`
  - `implies(vBatIndex(bn,fn)=i,`
    `vBatStart(bn,s1)=vFacStart(fn,i))`
  - `distribute` expects an <u>array</u> of variables?
  - Use multiple `occurrence`s instead

Copyright © Opta Consulting Ltd 2010

Implementation in Kalis 2

- Have to use main model/submodels
- Not intelligent about optimisation
  - `vLate(bn) + vOverdue(bn) >= vBatEnd(bn,"PAC") - iDueTime(bn)`
  - `vOver2(bn) = vBatEnd(bn,"PAC") - iDueTime(bn)`
  - `vOverdue(bn) = maximum({vOver2(bn),vZero})`
- Single step, 6 batches, did not finish

Copyright © Opta Consulting Ltd 2010

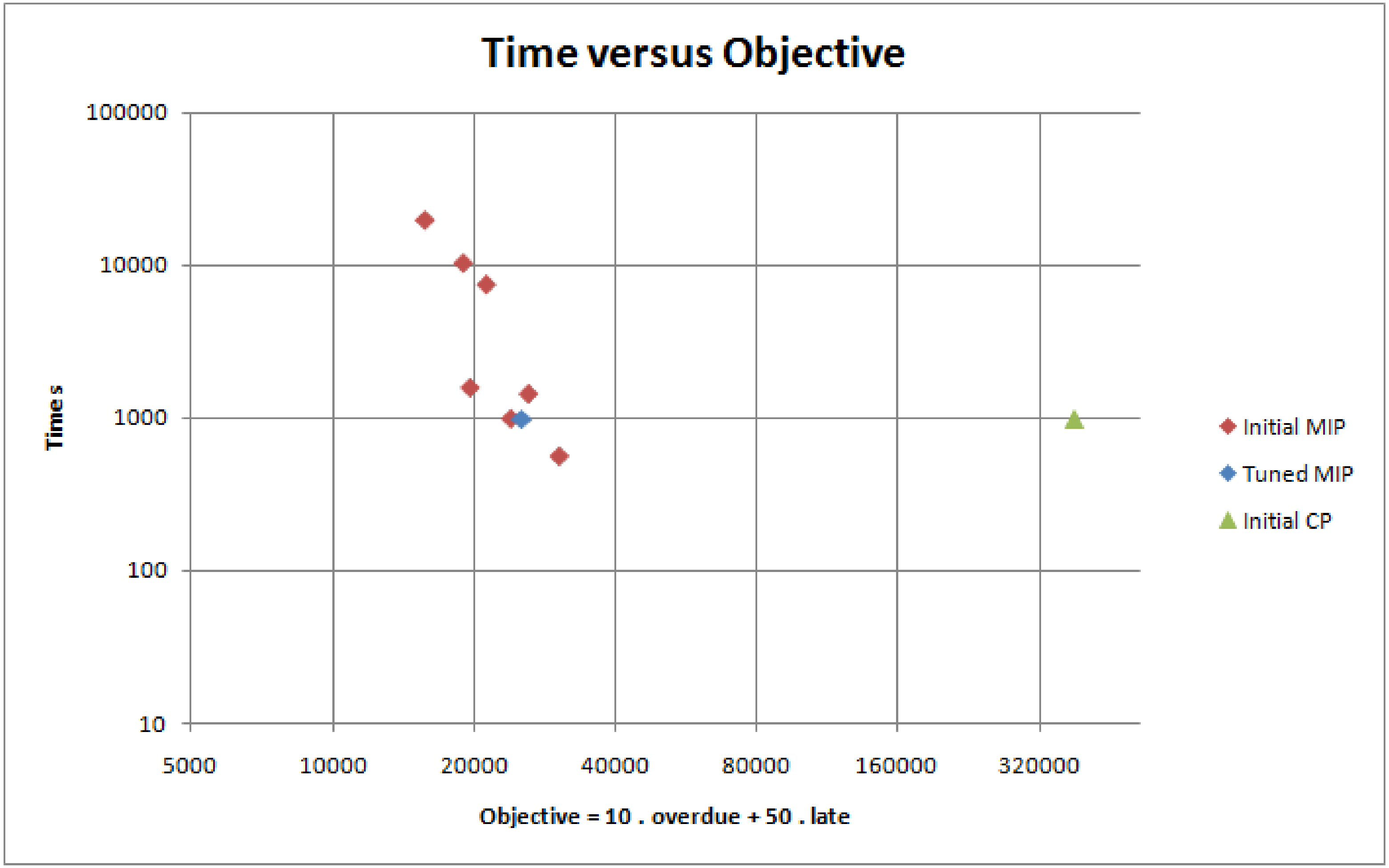
Implementation in Kalis 3

- Vital to specify strategy as well
  - Something like priorities in MIP …

```
cpbStrat(1) := assign_var(KALIS_INPUT_ORDER,
  KALIS_MAX_TO_MIN, vBatRun)
VarSet := {}
forall (bn in BatFlex)
             VarSet += {vBatFac(bn,"DIS")}
cpbStrat(2) :=
  assign_and_forbid(KALIS_SMALLEST_DOMAIN,
             KALIS_RANDOM_VALUE, VarSet)
```

Copyright © Opta Consulting Ltd 2010

First full CP Result



Copyright © Opta Consulting Ltd 2010

Kalis Enhancement 1

- Analysis of search tree
  - Zoom by selection rectangle
  - Branch path by double click
- Almost all time exploring equivalent mixer assignments
- So shortcut by fixing mixers in code
- Greedy algorithm based on disperser assignments is (almost) optimal
- But how to control call?

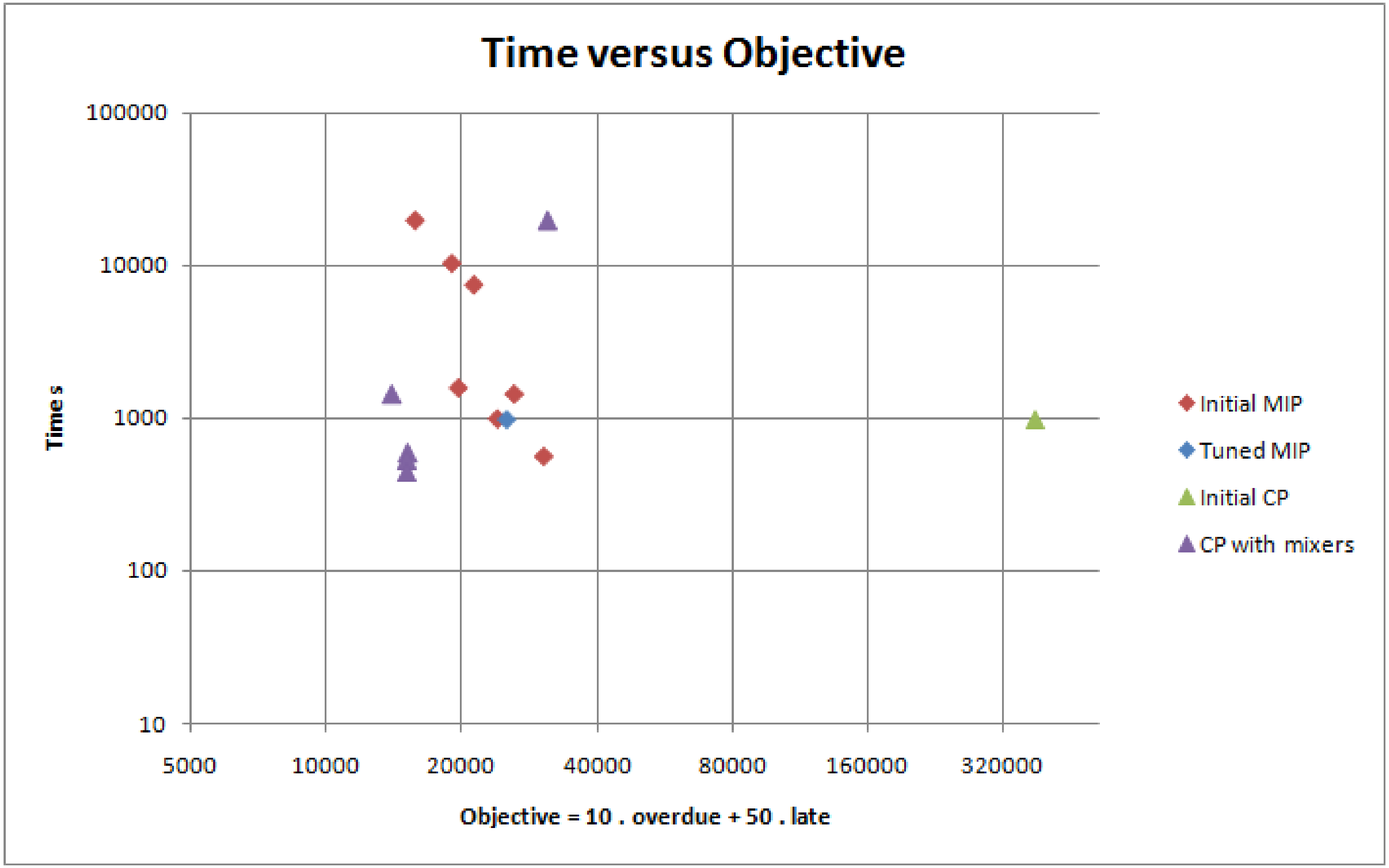
Copyright © Opta Consulting Ltd 2010

Kalis Enhancement 2

- Call routine as a user-defined strategy ... after disperser strategies
- Fix mixer variables with `setval`
- End with `cp_propagate` (OK if fail)
- Do not return a branching variable
- Must not run again in a descendent node
- Compare current depth to mix fix depth
- Have to create callbacks to know what depth is! (`cp_set_branch_callback`)

Copyright © Opta Consulting Ltd 2010

Improved CP Result



Copyright © Opta Consulting Ltd 2010

Fixing Mixers in MIP 1

- Would the same approach work in MIP?
- Apply as a cut manager callback
  - Use `setlb` and `setub`
  - Bounds are carried down tree
- When to apply – not automatic
  - Check for all dispersers at integer values
  - Cannot wait for all dispersers fixed
- Use auxiliary variable as a flag
  - Must have (tiny) positive entry in objective
  - Bound up to indicate mixers fixed - `getlb`

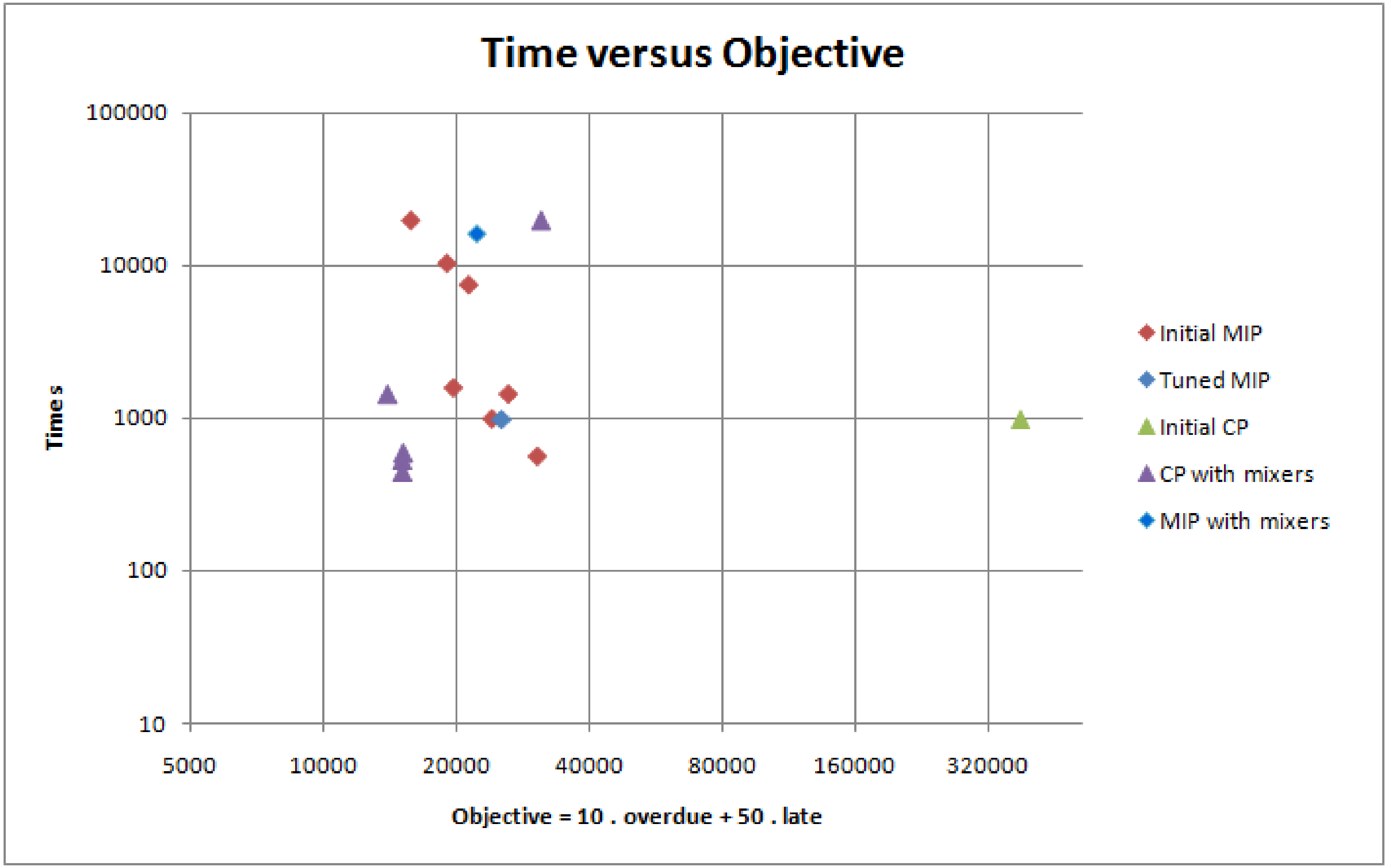
Copyright © Opta Consulting Ltd 2010

Fixing Mixers in MIP 2

- Errors!
  - Variables no longer existing
  - Conflicting bounds set already
  - Variables changing identity?
- Recommendation with custom cuts:
  - No presolve
  - No MIP presolve
  - No heuristics
  - No cut generation
- Observation – problems caused by:
  - Presolve (any options, even everything switched off)
  - MIP presolve - reduced cost fixing

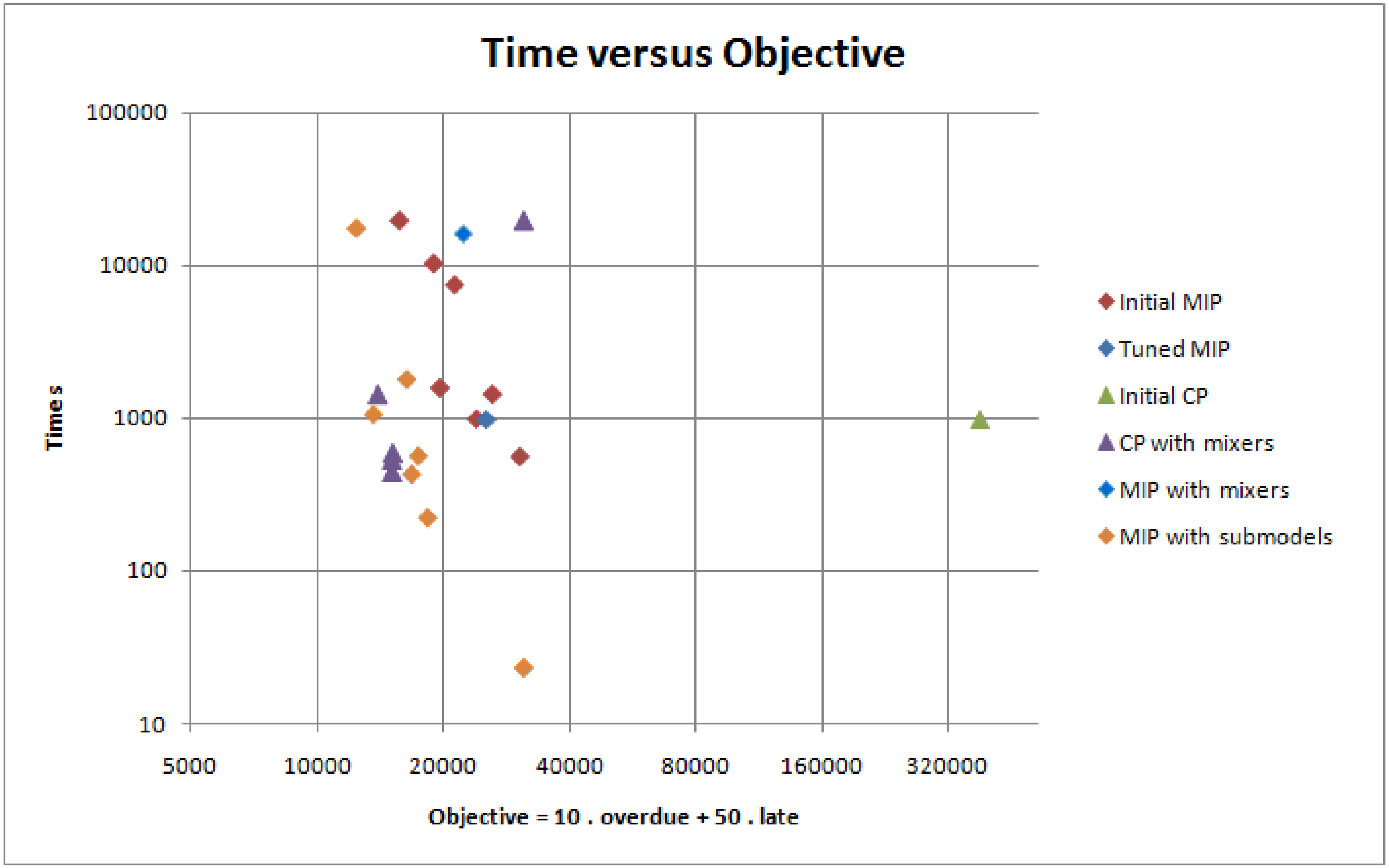
Copyright © Opta Consulting Ltd 2010

MIP with Fixed Mixers



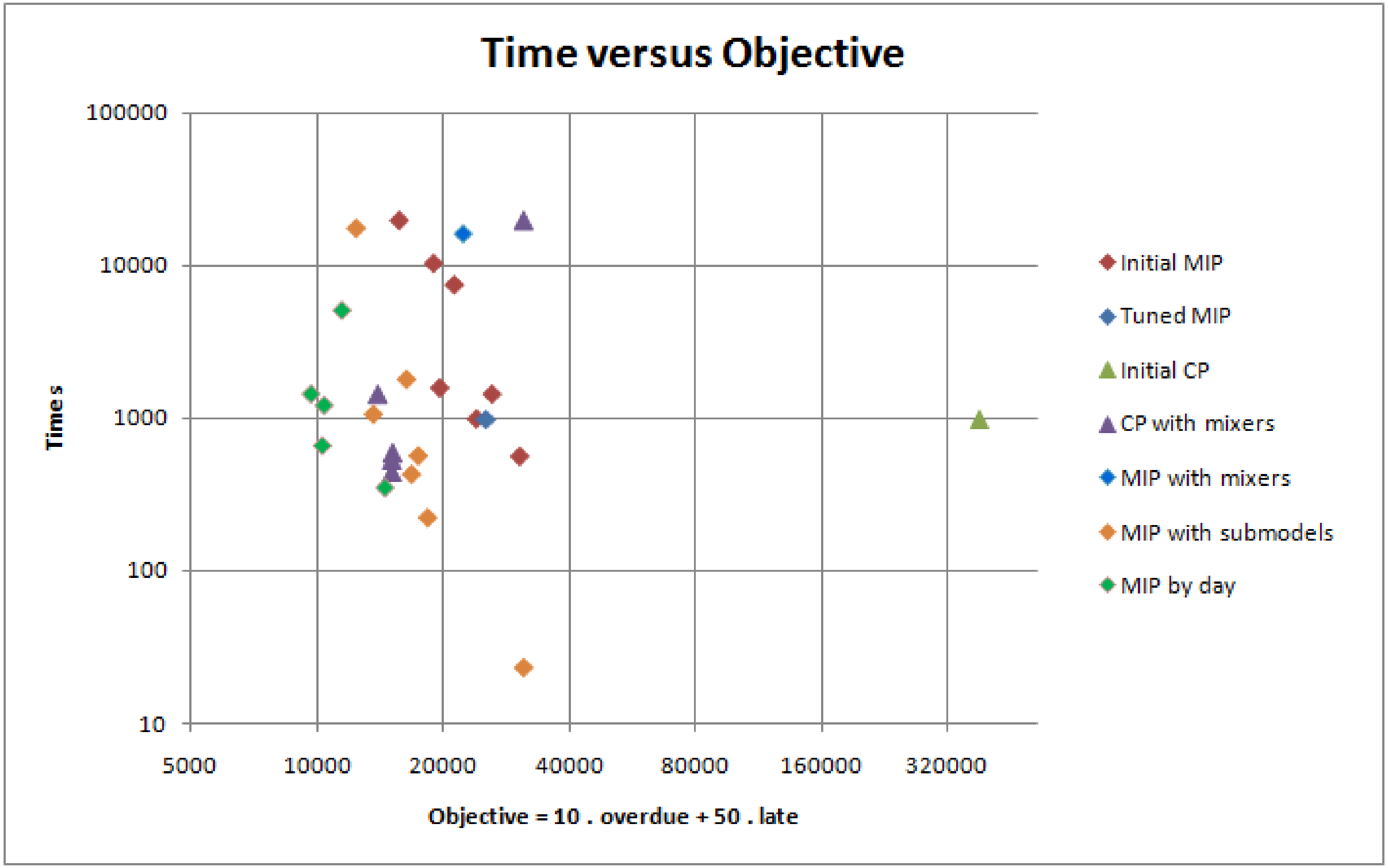
Copyright © Opta Consulting Ltd 2010

MIP with Submodels



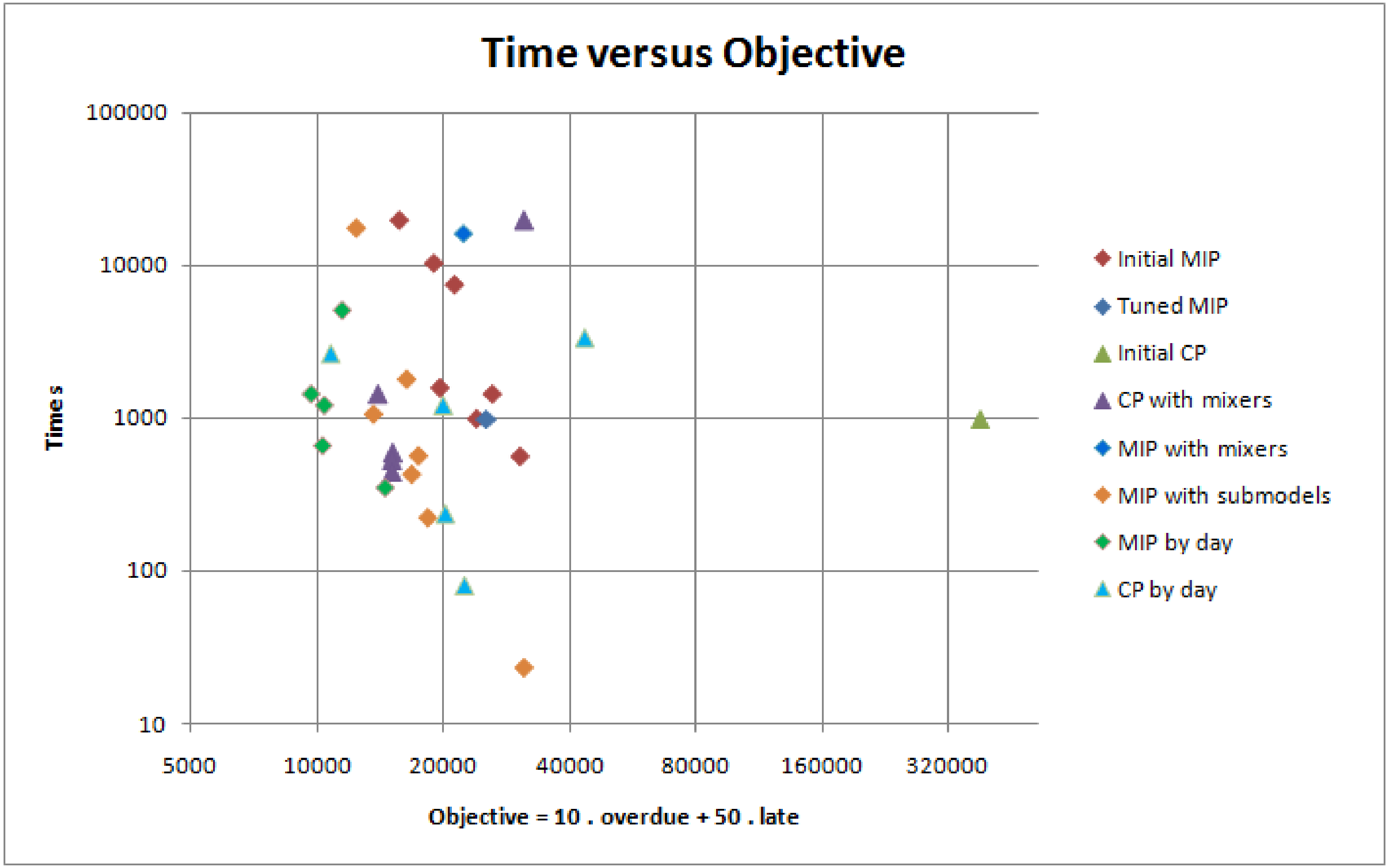
Copyright © Opta Consulting Ltd 2010

MIP Solving by Day



Copyright © Opta Consulting Ltd 2010

CP Solving by (Partial) Day

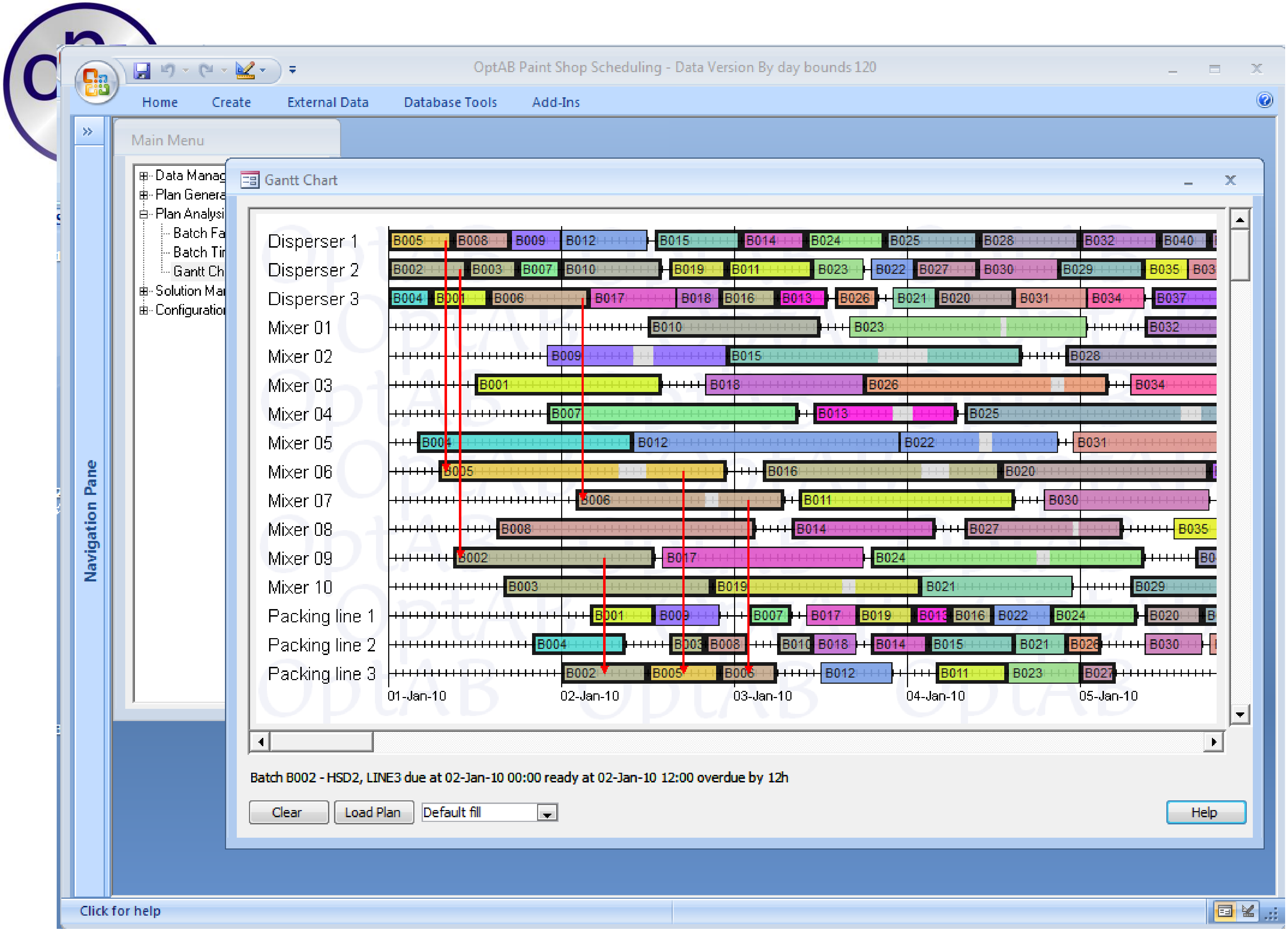


Copyright © Opta Consulting Ltd 2010

Conclusions

- CP
  - Conceptually easier but traps for the unwary
  - Kalis less well integrated in Mosel
  - Custom strategy vital to get results
  - Exponential time means exponential
- MIP
  - Decent solutions without tuning/customisation
  - Much better than exponential in practice
  - Custom algorithms face disadvantages
  - Hidden complexity – unexpected behaviour

Copyright © Opta Consulting Ltd 2010



Copyright © Opta Consulting Ltd 2010