OPTIMIZATION SOLUTIONS

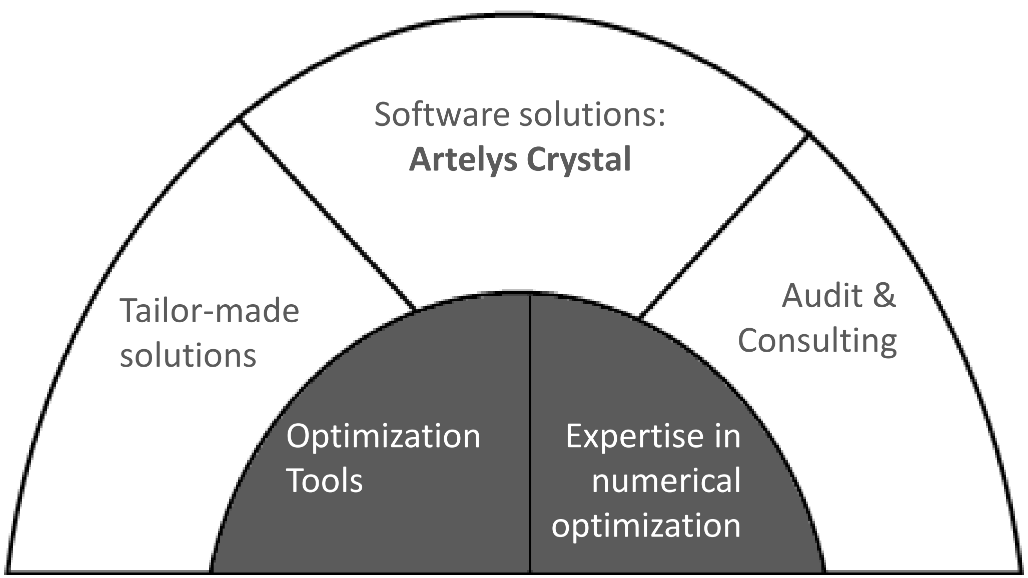
# Artelys: Optimization solutions

www.artelys.com

# Artelys: optimization & analytics

## Some figures

- Founded in 2000
- + 65% turnover growth between 2006 and 2012
- Team of 40 experts in optimization (engineers and PhDs)




## Artelys

- Specialized in **numerical optimization, statistics and decision-support** to solve large complex business problems
- Our core competences
  - **Numerical optimization and decision-support**
  - Consulting services and software

# Expertise in optimization & complex systems modeling

**Artelys experts support its clients in handling their complex problems:**

- Support in the usage of our optimization tools
  - Solver tuning to get the best performance
  - Bugs fixing

- Consulting in modeling & strategic optimization
  - Audit of optimization codes
  - Modeling support

- Trainings in
  - Numerical optimization
  - Statistical analysis
  - Modeling

# The most efficient **Optimization Tools**

## AMPL

- **Powerful algebraic modeling language** for linear and nonlinear optimization problems, with discrete or continuous variables
- Ideal for rapid prototyping and efficient use in production
- Best-in-class model presolver and automatic differentiator

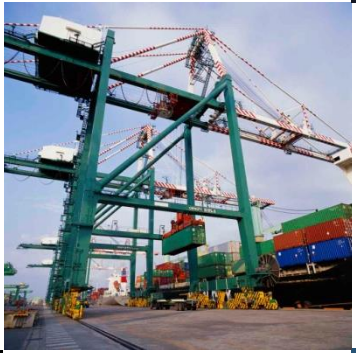
## KNITRO

- **Nonlinear programming and much more…**
- Active-set and interior-point/barrier algorithms for continuous optimization
- MINLP algorithms and complementary constraints for discrete optimization
- Parallel multi-start method for global optimization of non-convex problems

# FICO Xpress Optimization Suite

# Applications

- **Xpress is used in virtually all business sectors**
  - Energy / Oil & Gas
  - Mining
  - Industry / Manufacturing
  - Transportation / Logistics
  - Marketing
  - Finance / Banking
  - Computational Economics
  - Healthcare



# Xpress: short introduction

- **Developed by Dash Optimization, acquired by FICO in 2008**
- **Key features**
  - Full-featured, complete and versatile suite of tool for optimization practitioners and optimization application builders
  - State-of-the-art modeling and programming language : Mosel
  - Three complementary solvers : Optimizer, NonLinear, Kalis
  - Deployment facilities : Insight business platform and FICO Cloud
- **Many supported interfaces asides from Mosel**
  - C/C++, Java, .NET, Visual Basic, Fortran
  - AMPL
  - MATLAB
- **Supported platforms**
  - Windows 32-bit, 64-bit
  - Linux 32-bit, 64-bit
  - Mac OS X 32-bit, 64-bit
  - Solaris
- **Widely used in academia and industry**

# Xpress technologies and services

- **Access world-class professionals of optimization**
  - **Ongoing** development of solver and modeling engines by FICO's and Artelys' experts
  - Addition of many extra features based on **customer feedbacks** or project requirements
  - Supported by Artelys' consultants (PhD-level) who are used to solving the most difficult problems and deploying enterprise-wide optimization solutions
- **Combines efficiency and robustness for all problem classes**
  - **Optimizer** solves problems of the following classes: LP, QP, QCQP, MIP, MIQP, MIQCQP
  - **NonLinear** solves problems of the following classes : LP, QP, QCQP, SOCP, NLP
  - **Kalis** solves problems of the following classes : CP, scheduling, hybrid MIP/LP/CP

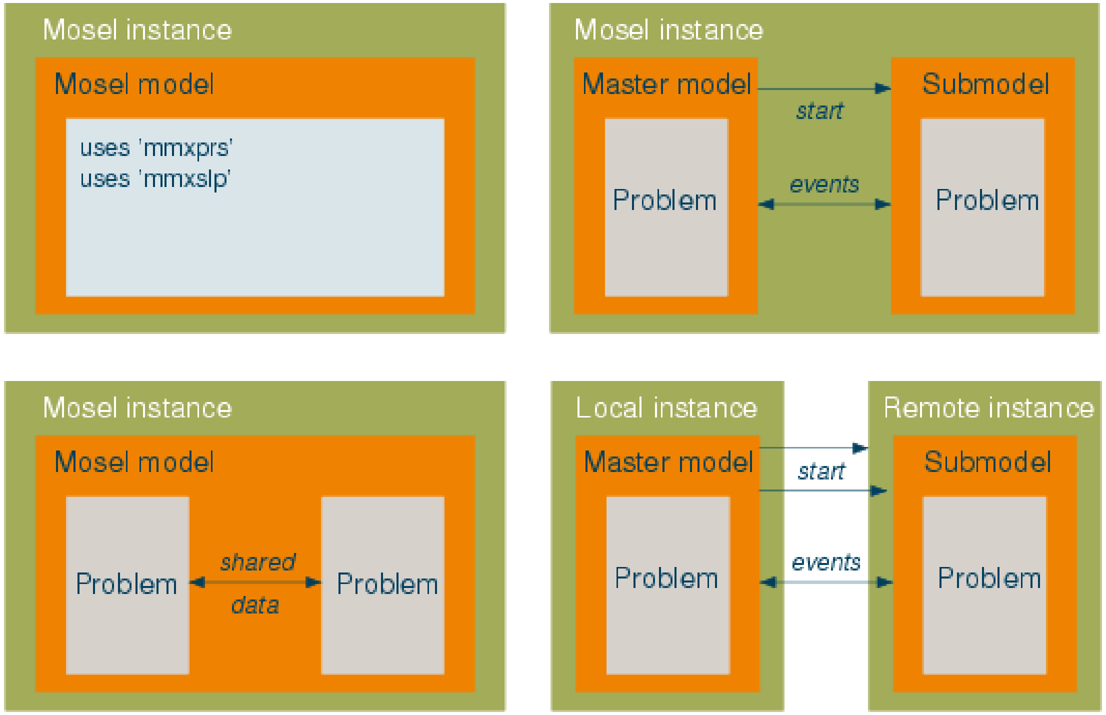
| Optimizer<br>NonLinear<br>Kalis | Mosel | Insight | FICO Cloud | Troubleshooting | Consultancy | Training |
|---|---|---|---|---|---|---|

| Development | Deployment | Professional support |
|---|---|---|

**Xpress technology and services**

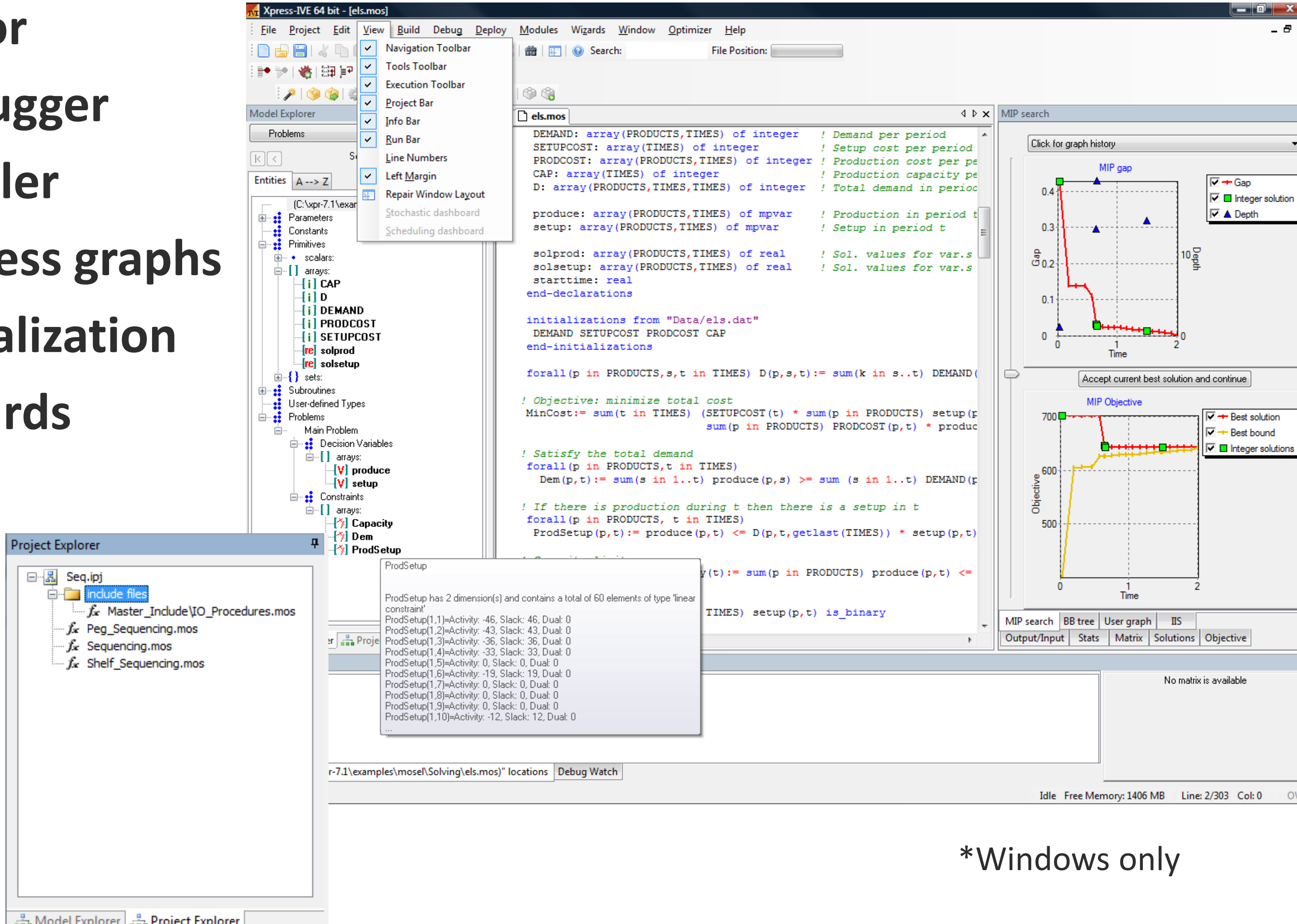
# Xpress-Mosel

- Concise and efficient programming language for optimization
- Enabled for distributed competing
- Provides connectors to ODBC databases, Oracle, Excel, Access, XML



# Xpress-IVE

- Editor
- Debugger
- Profiler
- Process graphs
- Visualization
- Wizards




*Windows only

# TSP problem

- **Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?**

- **$G(V,E)$ being the complete graph, $c_e$ is the cost of each edge $e = (i,j), i < j$, the TSP can be formulated as:**

$$
\begin{array}{lll}
\min & \sum_e c_e x_e & \\
\mid \quad (1) & x(\delta(v)) = 2 & \forall v \in V \\
\quad\;\; (2) & x(\delta(S)) \geq 2 & \forall S \subset V, \emptyset \neq S \neq V \\
\mid \quad x_e \in \{0;1\} & &
\end{array}
$$

- **There is $n!$ constraints (2), we will add them iteratively**

Artelys OPTIMIZATION SOLUTIONS

# Set up your environment

- **Compile the code**
  - git clone https://github.com/klorel/eps2016.git
  - go to the eps2016 directory
  - mkdir build and go to build
  - cmake ..
  - make
  - cd bin
  - eps2016_mip ../../data/eil51.txt 0

- **Launch eclipse and follow me**

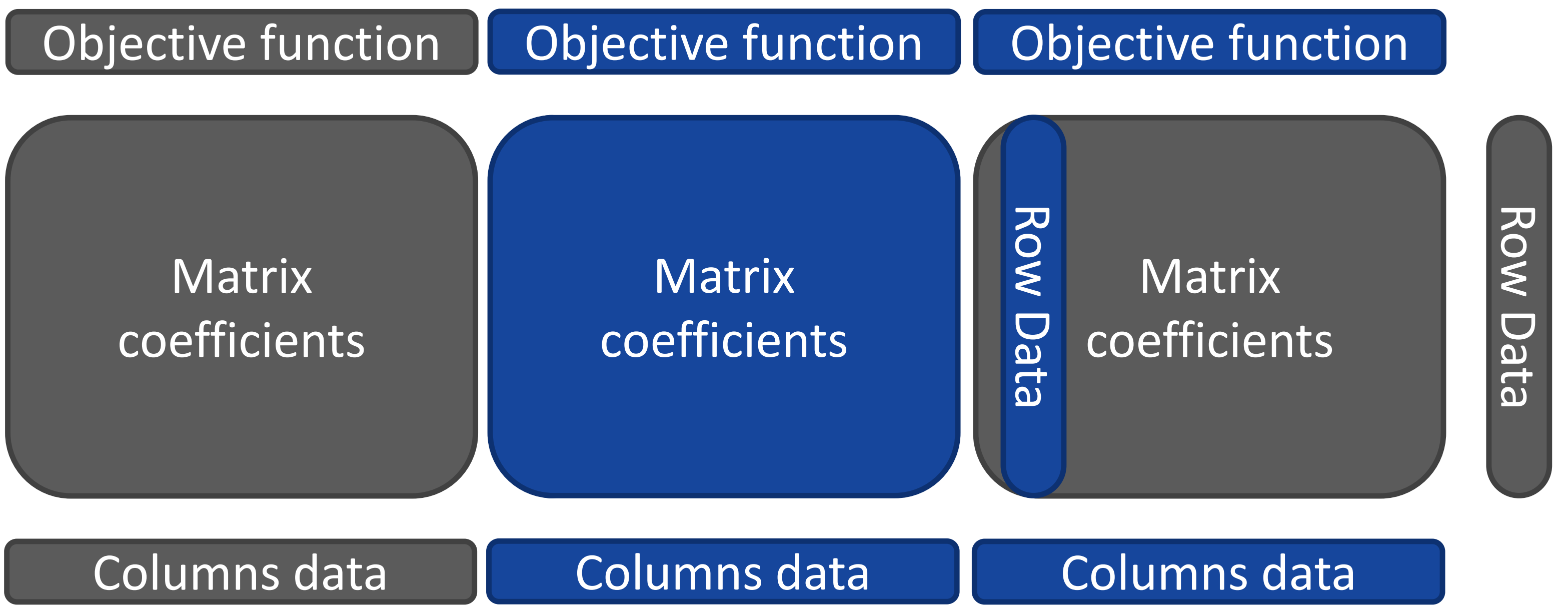
Artelys | OPTIMIZATION SOLUTIONS

# Optimization technics with FICO Optimizer

# Problem construction

- **What is a linear problem ?**

- **How to build it efficiently ?**
  - Rows and then columns
  - Columns and then rows



# Problem construction

- **Right Hand Side and row sense definition (see XPRSchgrhsrange)**

| Value of r | Row type | Effect |
|---|---|---|
| $r \geq 0$ | $= b, \leq b$ | $b - r \leq \sum a_j x_j \leq b$ |
| $r \geq 0$ | $\geq b$ | $b \leq \sum a_j x_j \leq b + r$ |
| $r < 0$ | $= b, \leq b$ | $b \leq \sum a_j x_j \leq b - r$ |
| $r < 0$ | $\geq b$ | $b + r \leq \sum a_j x_j \leq b$ |

- **Always try to mutualize call to the XPRS API function**
  - XPRSaddCols(…), XPRSaddRows(…), XPRSaddCuts(…)
  - XPRSchgbounds(), XPRSchgcoltype(), XPRSchgobj(), XPRSchgcoeff(…)

# Sparse matrix representation

- **For complex implementation the use of low level sparse data structure is the most efficient**

| By Rows | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | | | | | | | | | | | | | |
| colind | | | | | | | | | | | | | |
| rowstart | | | | | | | | | | | | | |

$$B = \begin{pmatrix} 1 & -1 & * & -3 & * \\ -2 & 5 & * & * & * \\ * & * & 4 & 6 & 4 \\ -4 & * & 2 & 7 & * \\ * & 8 & * & * & -5 \end{pmatrix}$$

| By Cols | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | | | | | | | | | | | | | |
| rowind | | | | | | | | | | | | | |
| colstart | | | | | | | | | | | | | |

# Sparse matrix representation

- **For complex implementation the use of low level sparse data structure is the most efficient**

| By Rows | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | 1 | -1 | -3 | -2 | 5 | 4 | 6 | -4 | 4 | 2 | 7 | 8 | -5 |
| colind | 0 | 1 | 3 | 0 | 1 | 2 | 3 | 4 | 0 | 2 | 3 | 1 | 4 |
| rowstart | 0 | 3 | 5 | 8 | 11 | 13 | | | | | | | |

$$B = \begin{pmatrix} 1 & -1 & * & -3 & * \\ -2 & 5 & * & * & * \\ * & * & 4 & 6 & 4 \\ -4 & * & 2 & 7 & * \\ * & 8 & * & * & -5 \end{pmatrix}$$

| By Cols | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | | | | | | | | | | | | | |
| rowind | | | | | | | | | | | | | |
| colstart | | | | | | | | | | | | | |

# Sparse matrix representation

- **For complex implementation the use of low level sparse data structure is the most efficient**

| By Rows | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | 1 | -1 | -3 | -2 | 5 | 4 | 6 | -4 | 4 | 2 | 7 | 8 | -5 |
| colind | 0 | 1 | 3 | 0 | 1 | 2 | 3 | 4 | 0 | 2 | 3 | 1 | 4 |
| rowstart | 0 | 3 | 5 | 8 | 11 | 13 | | | | | | | |

$$B = \begin{pmatrix} 1 & -1 & * & -3 & * \\ -2 & 5 & * & * & * \\ * & * & 4 & 6 & 4 \\ -4 & * & 2 & 7 & * \\ * & 8 & * & * & -5 \end{pmatrix}$$

| By Cols | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| values | 1 | -2 | -4 | -1 | 5 | 8 | 4 | 2 | -3 | 6 | 7 | 4 | -5 |
| rowind | 0 | 1 | 3 | 0 | 1 | 4 | 2 | 3 | 0 | 2 | 3 | 2 | 4 |
| colstart | 0 | 3 | 6 | 8 | 11 | 13 | | | | | | | |

# General considerations

- **The XPRESS presolve is very efficient and reduces a lot the problem**
  - Detection of redundant constraints
  - Bounds tightening using constraint propagation
  - Other options available : XPRS_PRESOLVEOPS

- **How to get the presolved problem ?**
  - Solve a problem limiting the number of iteration and ask for bounds
  - Useful to identify variables are fixed or to get tightened bounds

- **How to force XPRESS to keep variables in the presolved problem**
  - Use XPRSloadsecurevecs, useful for cutting plane or Benders algorithms

# TSP example

- **Get the C++ material and provide a, implementation of the formulation of the TSP**

$$
\begin{aligned}
&\min && \textstyle\sum_{i,j} c_{ij} x_{ij} \\
&(outFlow) && \textstyle\sum_{j \neq v} x_{vj} = 1 \quad \forall v \in V \\
&(inFlow) && \textstyle\sum_{i \neq v} x_{iv} = 1 \quad \forall v \in V \\
&x_{ij} \geq 0
\end{aligned}
$$

- **Build the problem using the C API of XPRESS**
  - XPRSloadlp, XPRSaddcols, XPRSaddrows

- **Give a name to columns and rows, export the problem to a .lp formatted file**
  - XPRSaddnames, XPRSwriteprob

- **Export the presolved problem to a .lp file**
  - XPRS_LPITERLIMIT=0, XPRS_DEFAULTALG=2

# Continuous Linear Programs

- **XPRESS provides two algorithms for continuous Linear Programs**
  - Simplex PRIMAL/DUAL (and the parallel dual simplex)
  - Barrierand CROSSOVER (why using crossover ?)

- **How to run the optimization ? XPRSminim(…) or XPRSmaxim(…)**

- **How to get the optimal solution ?**
  - Use XPRSgetlptol(…) to get the primal solution, the rows activity, the dual solution and the reduced cost

- **Efficient warmstart procedure for the simplex algorithm**
  - XPRSgetbasis(…), XPRSloadbasis(…)
  - Warmstart is efficient when using
    - primal and only modifying the objective
    - dual and only modifying the right hand side
  - Warmstart in dual might also be useful when adding constraint to the problem

- **XPRSloadbasis documentation:**
  - If the problem has been altered since saving an advanced basis, you may want to alter the basis as
  - follows before loading it
    - Make new variables non-basic at their lower bound (cstatus[icol]=0), unless a variable has an infinite lower bound and a finite upper bound, in which case make the variable non-basic at its upper bound
    - Make new constraints basic
    - Try not to delete basic variables, or non-basic constraints.

- **Callbacks available for continuous optimization are only related to log**
  - XPRSaddcbmessage, XPRSaddcbbariteration, XPRSaddcbbarlog, XPRSaddcblplog

# TSP example

- **Solve the problem, and get the solution, do you have an integer solution ? Try different LP algorithm**
  - XPRS_DEFAULTALG
- **Extract the basis, is it degenerated ?**
  - XPRSgetbasis
- **Swap variables in (but zero) and out of the basis and solve it again with the primal or dual algorithm, how many iterations are made ?**
  - XPRSloadbasis, XPRS_SIMPLEXITER
- **Is there sub tours in the solution ?**

# MIP

- **A MIP is defined by using in the problem columns which are not continuous**
  - C indicates a continuous column
  - B indicates a binary column
  - I indicates an integer column

- **Advanced presolve available for MIP problems**
  - SYMMETRY: try to detect symmetry in the problem and break them. Why is it useful ?
  - MIPPRESOLVE : additional presolve used at each nodes
  - PREPROBING : additional presolve fixing integer at values and see implications (constraint programming technics)

- **Resolution if launch using XPRSminim or XPRSmipoptimize**
  - Each integer solution can be retrieved looping over the solution pool

- **Sensitive analysis:** *no dual variables available***!**
  - fixGlobals and then get dual values for the fixed problem.

- **Stopping criterion, numerical parameters**
  - MIPTOL : tolerance used to declare a value is an integer
  - MIPRELGAP, MIPABSGAP : relative and absolute MIP gap (ub-lb)
  - MIPCUTOFF : artificial lower bound provided by the user
  - NODELIMIT, MAXTIME (CPUTIME), :

# TSP example

- **Change the column types to have them binary and solve the MIP relaxation with no sub tour constraint**
  - Get the number of nodes, the final gap.

- **Implement the sub tour breaking constraint for the new formulation**
  - $\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \forall \emptyset \neq S \neq V$

- **Use the iterative algorithm of TSAlgo to solve the new TSP formulation**

# Treatment of a node

- **The treatment of each node is basically a loop**
  - Presolve
  - Resolution of the relaxation
  - Cut generation
  - Heuristics to obtain a feasible solution

- **When ending this, a variable is selected, two nodes are created and added to the tree**

- **Dedicated parameters allow the user to tune the XPRESS behavior at the root node and within the B&B tree**
  - HEURSEARCHROOTSELECT and HEURSEARCHTREESELECT
  - CUTSELECT and TREECUTSELECT
  - PRESOLVE and MIPPRESOLVE, TREEPRESOLVE

- **Several combination can be automatically determined by the XPRESS tuner (only on windows)**

# TSP example

- **Disable the cutting phase or the heuristic phase, increase their effort?**
  - Look at the number of nodes (XPRS_NODES) performed during the optimization process

- **Try several values for the root and tree parameters of OPTIMIZER**
  - HEURSEARCHROOTSELECT and HEURSEARCHTREESELECT
  - CUTSELECT and TREECUTSELECT
  - PRESOLVE and MIPPRESOLVE, TREEPRESOLVE

# MIP callbacks

- **MIP callback ban be use to interact with the solver within the B&B, the most useful are**
  - optnode : after the relaxation has been solved
  - preIntsol : each time an integer solution is found
  - Intsol : each time an integer solution is accepted

- **Other callbacks:**
  - Nodecutoff, Chgbranch, Infnode, Chgbounds, Prenode, Newnode, Chgnode, cutmgr

- **A callback is a function with a given prototype, see XPRSaddcbXXX to see detailed information of callback XXX.**

```
int XPRS_CC XPRSaddcboptnode(
        XPRSprob prob,
        void (XPRS_CC *f_optnode)(XPRSprob my_prob, void *my_object, int *feas),
        void *object, int priority
);
```

- **XPRSaddcboptnode allows the user to define the a callback that will be called after each relaxation resolution**
  - *my_object* can be used to pass user defined data structure, it will be available at each call of the callback function

# TSP example

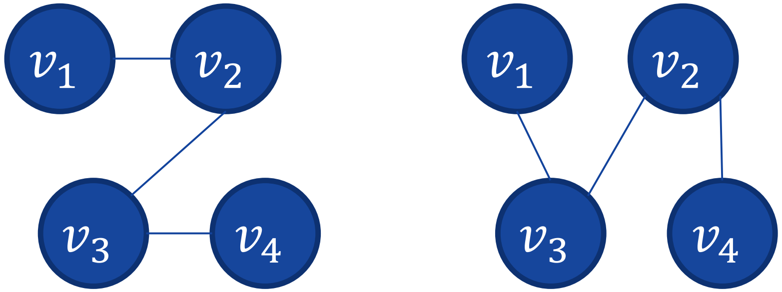
- **Use the run_callback method of the TSPAlgo to launch a resolution where cuts are added within the optnode callback**
  - Use XPRSgetlpsol() to get the optimal solution
  - The MIPINFEAS attribute returns the number of integer infeasibilities at the current nodes
  - Need to store the already added cuts

- **Observe that this time the B&B converge to a solution without sub tours. How many cuts are added ?**

- **Modify the code to only add cuts associated with the smallest sub tour**

# TSP example

- **Use the intsol to use a local search algorithm performing all possible inversion a tour**
  - 1-2-3 gives 2-1-3, 1-3-2, 3-1-2, etc.
  - Any improved solution can be transfer to XPRESS using XPRSaddmipsol
  - $v_1 - v_2 - v_3 - v_4$ can be improved in $v_1 - v_3 - v_2 - v_4$ iff
    - $e_{v_1 v_2} + e_{v_3 v_4} > e_{v_1 v_3} + e_{v_2 v_4}$




- **Try other moves**

# TSP example

- **Use your MIP solver to implement a VNS based heuristic with XPRESS optimizer**
  - Given a integer solution, chose a node $v_0$ and solve the TSP induced by optimizing the $k$ neighbors of $v_0$
  - If the solution is improved $k = 0$ else $k += 1$
  - If $k$ exceeds $kMax$, $k = 1$

- **How to fix variables ?**
  - fixGlobal can be used to fix all integer variables and then to optimize the resulting continuous integer programming
  - A subset of integer variables can be fixed/unfixed using the XPRSchgbounds (with binary bounds values)

- **Advanced selection of a nodes ?**
  - Get the dual values of the degree constraints and try to use them within the node selection processus