Artelys | OPTIMIZATION SOLUTIONS

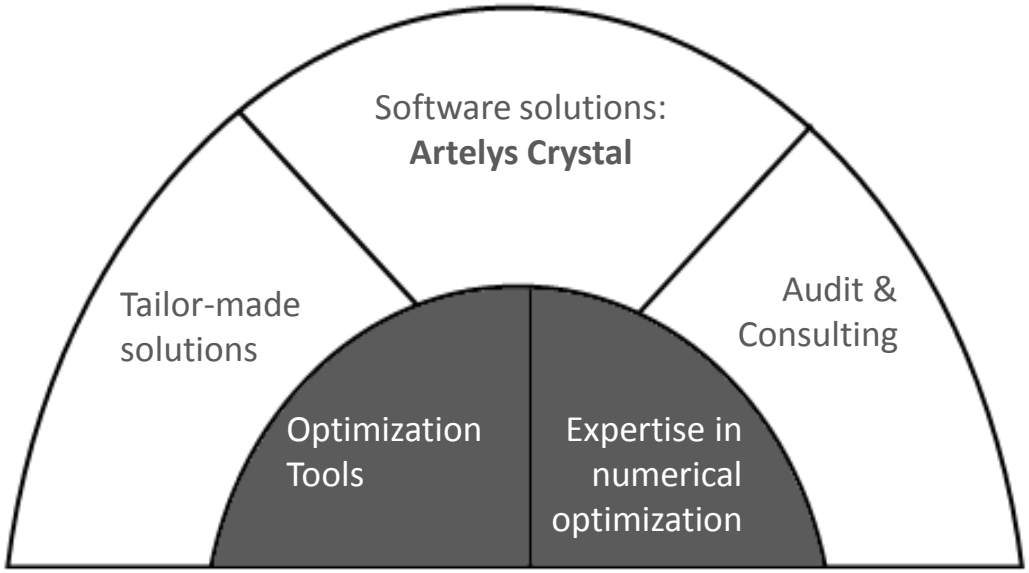
Artelys: Optimization solutions

www.artelys.com

Artelys: optimization & analytics

Some figures

- Founded in 2000
- + 65% turnover growth between 2006 and 2012
- Team of 30 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

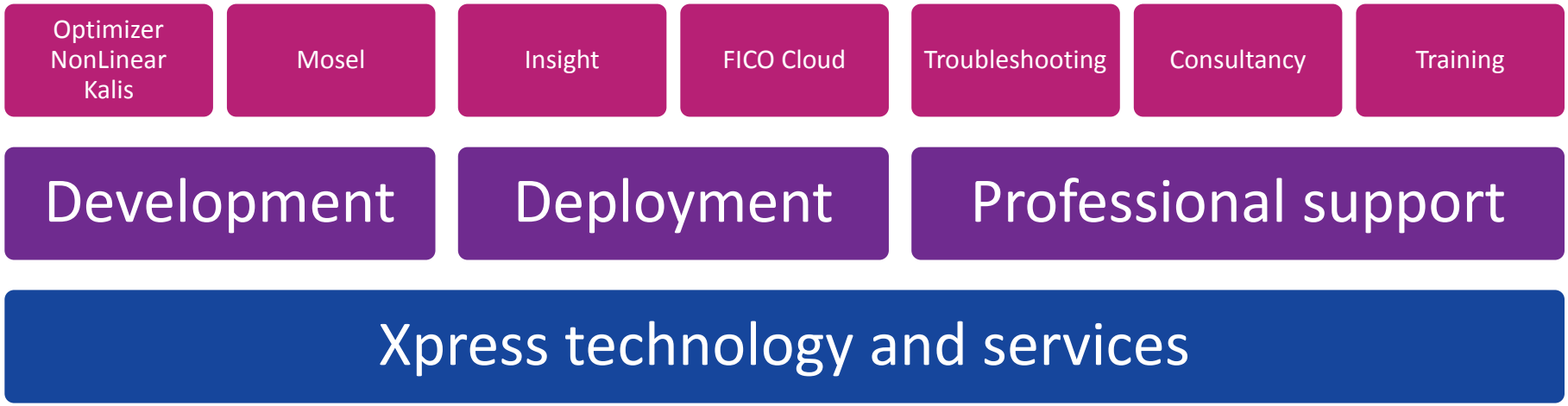
Xpress: short introduction

- **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**

Artelys | OPTIMIZATION SOLUTIONS

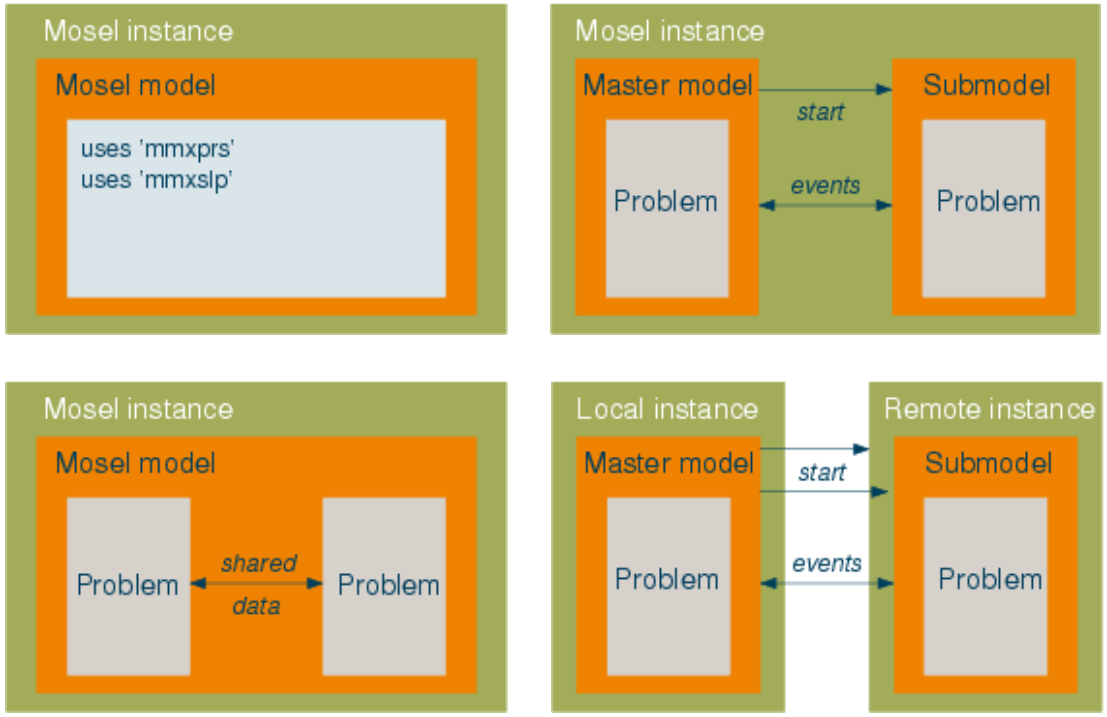
Xpress technologies and services

- **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



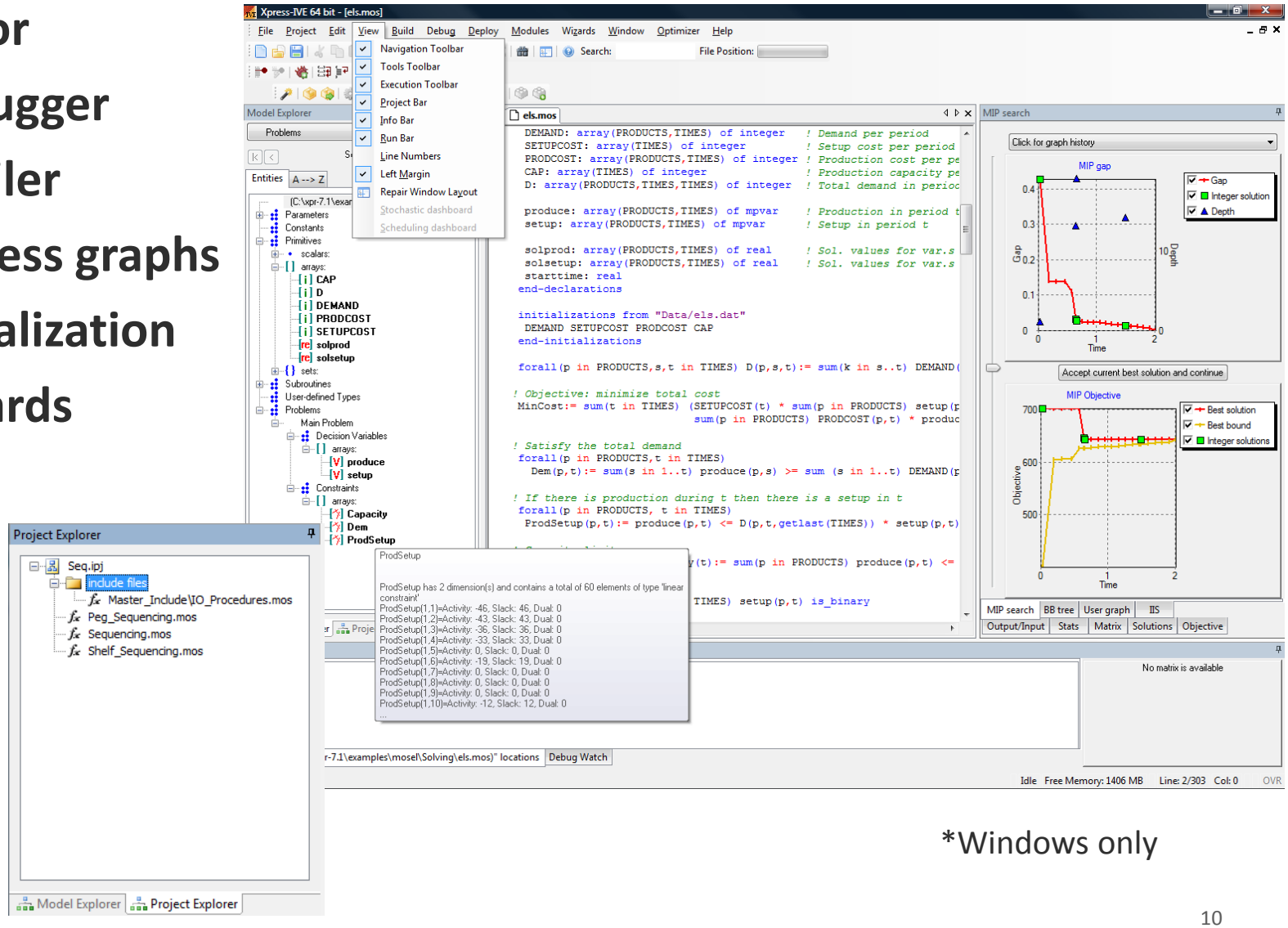
Xpress-Mosel

- 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?**

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

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

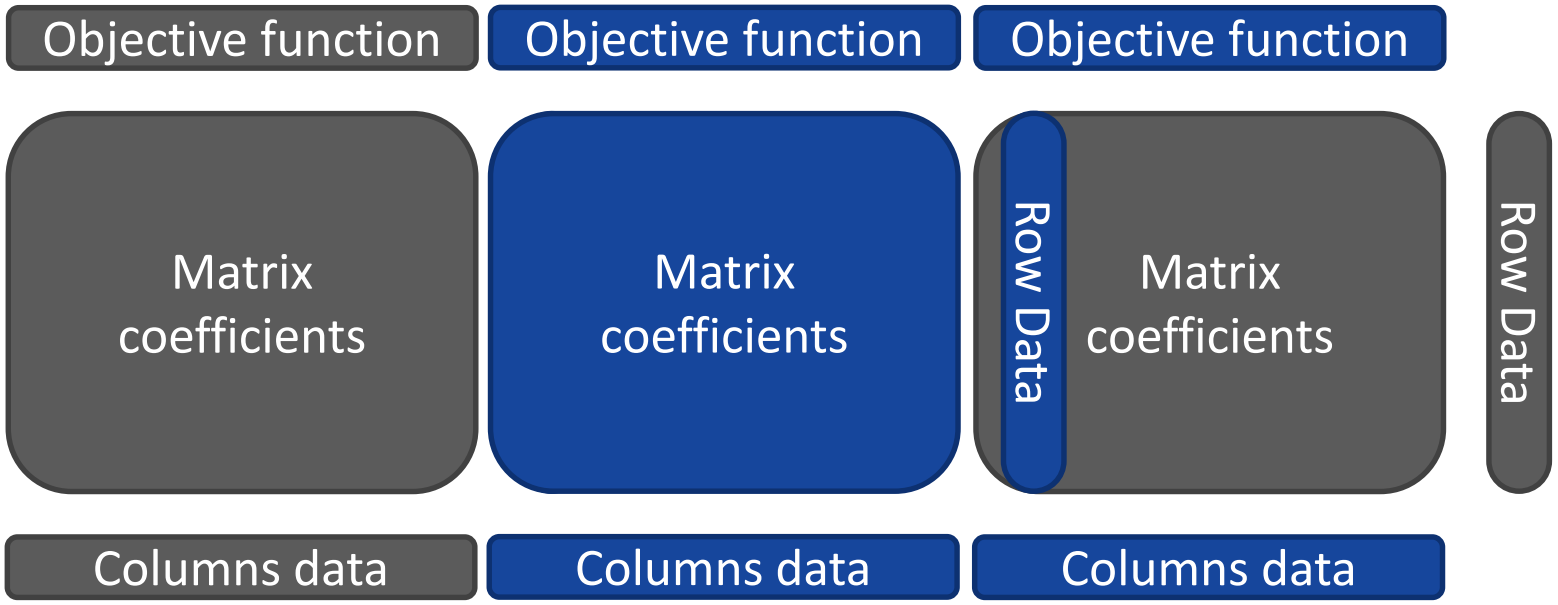
- **There is $n!$ constraints (2), we will add them iteratively**

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													

By Cols													
values													
rowind													
colstart													

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

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							

By Cols													
values													
rowind													
colstart													

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

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							

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							

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

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

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

- **Get the c++ material and build the lp relaxation of the TSP problem where (2) is relaxed**

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

- **Solve the problem, and get the solution, do you have an integer solution ?**

- **Use the Xprschgbounds to fix all variables to a given integer solution ?**
  - Try to build up a feasible solution satisfying (2), ie no sub tours

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

TSP example

- **Change the column types to have them binary and solve the MIP relaxation**

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

- **Try to detect the subtour in the solution ?**
  - Simple algorithm performing aggregations of subtours

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)

- **Add the constraint breaking one/all the subtour and resolve to see the subtour was broken.**

- **Build up a loop iterating until there is no more sub tour in the solution**

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
);
```

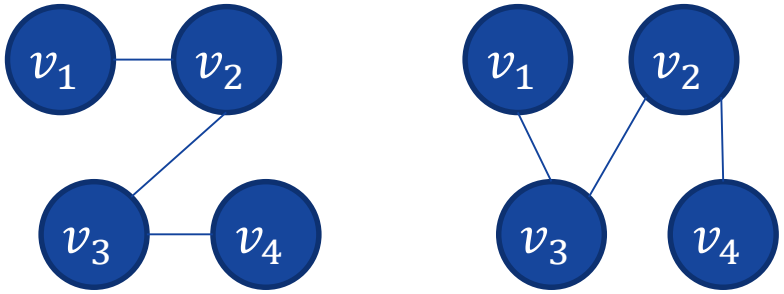
- **XPRSaddcboptnode allows the user to define the a callback that will be called after each relaxation resolution**
  - *my_object* can be used to pass user defined data structure, it will be available at each call of the callback function

TSP example

- **Use the Optnode to add the subtour breaking constraints within the B&B algorithm**
  - Use XPRSgetlpsol() to get the optimal solution
  - The MIPINFEAS attribute returns the number of integer infeasibilities at the current nodes
- **Observe that this time the B&B converge to a solution without sub tours**

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
  - Disable heuristics and cuts generation to make you're the solution is found with your method
  - $v_1 - v_2 - v_3 - v_4$ can be improved in $v_1 - v_3 - v_2 - v_4$ iff
    - $e_{v_1 v_2} + e_{v_3 v_4} > e_{v_1 v_3} + e_{v_2 v_4}$



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 $k \mathrel{+}= 1$
  - 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