

Branch and Bound

The FICO Xpress Optimizer uses the approach of LP based branch and bound with cutting planes for solving Mixed Integer Programming (MIP) problems. That is, the Optimizer solves the optimization problem (typically an LP problem) resulting from relaxing the discreteness constraints on the variables and then uses branch and bound to search the relaxation space for MIP solutions. It combines this with heuristic methods to quickly find good solutions, and cutting planes to strengthen the LP relaxations.

The Optimizer's MIP solving methods are coordinated internally by sophisticated algorithms so the Optimizer will work well on a wide range of MIP problems with a wide range of solution performance requirements without any user intervention in the solving process. Despite this the user should note that the formulation of a MIP problem is typically not unique and the solving performance can be highly dependent on the formulation of the problem. It is recommended, therefore, that the user undertake careful experimentation with the problem formulation using realistic examples before committing the formulation for use on large production problems. It is also recommended that users have small scale examples available to use during development.

Because of the inherent difficulty in solving MIP problems and the variety of requirements users have on the solution performance on these problems it is not uncommon that users would like to improve over the default performance of the Optimizer. In the following sections we discuss aspects of the branch and bound method for which the user may want to investigate when customizing the Optimizer's MIP search.

THEORY

In this section we present a brief overview of branch and bound theory as a guide for the user on where to look to begin customizing the Optimizer's MIP search and also to define the terminology used when describing branch and bound methods.

To simplify the text in the following, we limit the discussion to MIP problems with linear constraints and a linear objective function. Note that it is not difficult to generalize the discussion to problems with quadratic constraints and a quadratic objective function.

The branch and bound method has three main concepts: relaxation, branching and fathoming.

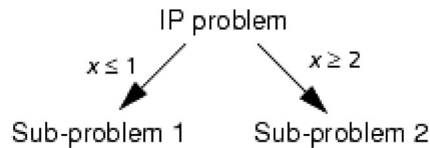
The relaxation concept relates to the way discreteness or integrality constraints are dropped or 'relaxed' in the problem. The initial relaxation problem is a Linear Programming (LP) problem which we solve resulting in one of the following cases:

- a. The LP is infeasible so the MIP problem must also be infeasible;
- b. The LP has a feasible solution, but some of the integrality constraints are not satisfied – the MIP has not yet been solved;
- c. The LP has a feasible solution and all the integrality constraints are satisfied so the MIP has also been solved;
- d. The LP is unbounded.

Case (d) is a special case. It can only occur when solving the initial relaxation problem and in this situation the MIP problem itself is not well posed (see Chapter *Infeasibility, Unboundedness and Instability* (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/chapter3.html>) for details about what to do in this case). For the remaining discussion we assume that the LP is not unbounded.

Outcomes (a) and (c) are said to 'fathom' the particular MIP, since no further work on it is necessary. For case (b) more work is required, since one of the unsatisfied integrality constraints must be selected and the concept of separation applied.

To illustrate the branching concept suppose, for example, that the optimal LP value of an integer variable x is 1.34 , a value which violates the integrality constraint. It follows that in any solution to the original problem either $x \leq 1.0$ or $x \geq 2.0$. If the two resulting MIP problems are solved (with the integrality constraints), all integer values of x are considered in the combined solution spaces of the two MIP problems and no solution to one of the MIP problems is a solution to the other. In this way we have branched the problem into two disjoint *sub-problems*.



If both of these sub-problems can be solved and the better of the two is chosen, then the MIP is solved. By recursively applying this same strategy to solve each of the sub-problems and given that in the limiting case the integer variables will have their domains divided into fixed integer values then we can guarantee that we solve the MIP problem.

Branch and bound can be viewed as a *tree-search* algorithm. Each *node* of the tree is a MIP problem. A MIP node is relaxed and the LP relaxation is solved. If the LP relaxation is not fathomed, then the node MIP problem is partitioned into two more sub-problems, or *child* nodes. Each child MIP will have the same constraints as the *parent* node MIP, plus one additional inequality constraint. Each node is therefore either fathomed or has two children or *descendants*.

We now introduce the concept of a *cutoff*, which is an extension of the fathoming concept. To understand the cutoff concept we first make two observations about the behavior of the node MIP problems. Firstly, the optimal MIP objective of a node problem can be no better than the optimal objective of the LP relaxation. Secondly, the optimal objective of a child LP relaxation can be no better than the optimal objective of its parent LP relaxation. Now assume that we are exploring the tree and we are keeping the value of the best MIP objective found so far. Assume also that we keep a 'cutoff value' equal to the best MIP objective found so far. To use the cutoff value we reason that if the optimal LP relaxation objective is no better than the cutoff then any MIP solution of a descendant can be no better than the cutoff and the node can be fathomed (or cutoff) and need not be considered further in the search.

The concept of a cutoff can be extended to apply even when no integer solution has been found in situations where it is known, or may be assumed, from the outset that the optimal solution must be better than some value. If the relaxation is worse than this cutoff, then the node may be fathomed. In this way the user can reduce the number of nodes processed and improve the solution performance. Note that there is the possibility, however, that all MIP solutions, including the optimal one, may be missed if an overly optimistic cutoff value is chosen.

The cutoff concept may also be extended in a different way if the user intends only to find a solution within a certain tolerance of the overall optimal MIP solution. Assume that we have found a MIP solution to our problem and assume that the cutoff is maintained at a value 100 objective units better than the current best MIP solution. Proceeding in this way we are guaranteed to find a MIP solution within 100 units of the overall MIP optimal since we only cutoff nodes with LP relaxation solutions worse than 100 units better than the best MIP solution that we find.

If the MIP problem contains SOS entities then the nodes of the branch and bound tree are determined by branching on the sets. Note that each member of the set has a double precision reference row entry and the sets are ordered by these reference row entries. Branching on the sets is done by choosing a position in the ordering of the set variables and setting all members of the set to 0 either above or below the chosen point. The optimizer used the

reference row entries to decide on the branching position and so it is important to choose the reference row entries which reflect the cost of setting the set member to 0. In some cases it may be better to model the problem with binary variables instead of special ordered sets. This is especially the case if the sets are small.

VARIABLE SELECTION AND CUTTING

The branch and bound technique leaves many choices open to the user. In practice, the success of the technique is highly dependent on several key choices.

- a. Deciding which variable to branch on is known as the *variable selection problem* and is often the most critical choice.
- b. *Cutting planes* are used to strengthen the LP relaxation of a sub problem, and can often bring a significant reduction in the number of sub-problems that must be solved

The Optimizer incorporates a default strategy for both choices which has been found to work adequately on most problems. Several controls are provided to tailor the search strategy to a particular problem.

VARIABLE SELECTION FOR BRANCHING

Each global entity has a priority for branching, or one set by the user in the directives file. A *low* priority value means that the variable is *more* likely to be selected for branching. The Optimizer uses a priority range of 400–500 by default. To guarantee that a particular global entity is always branched first, the user should assign a priority value less than 400. Likewise, to guarantee that a global entity is only branched on when it is the only candidate left, a priority value above 500 should be used.

The Optimizer uses a wide variety of information to select among those entities that remain unsatisfied and which belong to the lowest valued priority class. A *pseudo cost* is calculated for each candidate entity, which is typically an estimate of how much the LP relaxation objective value will change (degradation) as a result of branching on this particular candidate. Estimates are calculated separately for the up and down branches and combined according to the strategy selected by the `VARSELECTION` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/VARSELECTION.html>) control.

The default strategy is based on calculating pseudo costs using the method of *strong branching*. With strong branching, the LP relaxations of the two potential sub problems that would result from branching on a candidate global entity, are solved partially. Dual simplex is applied for a limited number of iterations and the change in objective value is recorded as a pseudo cost. This can be very expensive to apply to every candidate for every node of the branch and bound search, which is why the Optimizer by default will reuse pseudo costs collected from one node, on subsequent nodes of the search.

Selecting a global entity for branching is a multi-stage process, which combines estimates that are cheap to compute, with the more expensive strong branching based pseudo costs. The basic selection process is given by the following outline, together with the controls that affect each step:

1. Pre-filter the set of candidate entities using very cheap estimates.
`SBSELECT` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/SBSELECT.html>): **determine the filter size.**
2. Calculate simple estimates based on local node information and rank the selected candidates.
`SBESTIMATE` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/SBESTIMATE.html>): **local ranking function.**
3. Calculate strong-branching pseudo costs for candidates lacking such information.
`SBBEST` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/SBBEST.html>): **number of variables to strong branch on.**

SBITERLIMIT (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/SBITERLIMIT.html>): LP iteration limit for strong branching.

4. Select the best candidate using a combination of pseudo costs and the local ranking functions.

The overall amount of effort put into this process can be adjusted using the SBEFFORT (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/SBEFFORT.html>) control.

CUTTING PLANES

Cutting planes are valid constraints used for tightening the LP relaxation of a MIP problem, without affecting the MIP solution space. They can be very effective at reducing the amount of sub problems that the branch and bound search has to solve. The Optimizer will automatically create many different well-known classes of cutting planes, such as *mixed integer Gomory cuts*, *lift-and-project cuts*, *mixed integer rounding (MIR) cuts*, *clique cuts*, *implied bound cuts*, *flow-path cuts*, *zero-half cuts*, etc. These classes of cuts are grouped together into two groups that can be controlled separately. The following table lists the main controls and the related cut classes that are affected by those control:

COVERCUTS (https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/COVERCUTS.html)	Mixed integer rounding cuts
TREECOVERCUTS (https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/TREECOVERCUTS.html)	Lifted cover cuts
	Clique cuts
	Implied bound cuts
	Flow-path cuts
	Zero-half cuts
GOMCUTS (https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/GOMCUTS.html)	Mixed integer Gomory cuts
TREEGOMCUTS (https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/TREEGOMCUTS.html)	Lift-and-project cuts

The controls COVERCUTS (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/COVERCUTS.html>) and GOMCUTS (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/GOMCUTS.html>) sets an upper limit on the number of rounds of cuts to create for the root problem, for their respective groups. Correspondingly, TREECOVERCUTS (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/TREECOVERCUTS.html>) and TREEGOMCUTS (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/TREEGOMCUTS.html>) sets an upper limit on the number of rounds of cuts for any sub problem in the tree.

An important aspect of cutting is the choice of how many cuts to add to a sub problem. The more cuts that are added, the harder it becomes to solve the LP relaxation of the node problem. The tradeoff is therefore between the additional effort in solving the LP relaxation versus the strengthening of the sub problem. The CUTSTRATEGY

(<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/CUTSTRATEGY.html>) **control** sets the general level of how many cuts to add, expressed as a value from 0 (no cutting at all) to 3 (high level of cuts).

Another important aspect of cutting is how often cuts should be created and added to a sub problem. The Optimizer will automatically decide on a frequency that attempts to balance the effort of creating cuts versus the benefits they provide. It is possible to override this and set a fixed strategy using the **CUTFREQ** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/CUTFREQ.html>) **control**. When set to a value k , cutting will be applied to every k 'th level of the branch and bound tree. Note that setting **CUTFREQ** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/CUTFREQ.html>) = 0 will disable cutting on sub problems completely, leaving only cutting on the root problem.

NODE SELECTION

The Optimizer applies a search scheme involving best-bound first search combined with dives. Sub problems that have not been fathomed or which have not been branched further into new sub problems are referred to as *active nodes* of the branch and bound search tree. Such activate nodes are maintained by the Optimizer in a pool.

The search process involves selecting a sub problem (or node) from this active nodes pool and commencing a dive. When the Optimizer branches on a global entity and creates the two sub problems, it has a choice of which of the two sub problems to work on next. This choice is determined by the **BRANCHCHOICE** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/BRANCHCHOICE.html>) **control**. The dive is a recursive search, where it selects a child problem, branches on it to create two new child problems, and repeats with one of the new child problems, until it ends with a sub problem that should not be branched further. At this point it will go back to the active nodes pool and pick a new sub problem to perform a dive on. This is called a *backtrack* and the choice of node is determined by the **BACKTRACK** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/BACKTRACK.html>) **control**. The default backtrack strategy will select the active node with the best bound.

ADJUSTING THE CUTOFF VALUE

The parameter **MIPADDCUTOFF** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MIPADDCUTOFF.html>) determines the cutoff value set by the Optimizer when it has identified a new MIP solution. The new cutoff value is set as the objective function value of the MIP solution plus the value of **MIPADDCUTOFF**. If **MIPADDCUTOFF** has not been set by the user, the value used by the Optimizer will be calculated after the initial LP optimization step as:

$$\max(\text{MIPADDCUTOFF}, 0.01 \cdot \text{MIPRELCUTOFF} \cdot LP_value)$$

using the initial values for **MIPADDCUTOFF** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MIPADDCUTOFF.html>) and **MIPRELCUTOFF** (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MIPRELCUTOFF.html>), and where *LP_value* is the optimal objective value of the initial LP relaxation.

STOPPING CRITERIA

Often when solving a MIP problem it is sufficient to stop with a good solution instead of waiting for a potentially long solve process to find an optimal solution. The Optimizer provides several stopping criteria related to the solutions found, through the MIPRELSTOP and MIPABSSTOP parameters. If MIPABSSTOP is set for a minimization problem, the Optimizer will stop when it finds a MIP solution with an objective value equal to or less than MIPABSSTOP. The MIPRELSTOP parameter can be used to stop the solve process when the found solution is sufficiently close to optimality, as measure relative to the best available bound. The optimizer will stop due to MIPRELSTOP when the following is satisfied:

$$| \text{MIPOBJVAL} - \text{BESTBOUND} | \leq \text{MIPRELSTOP} \cdot \max(| \text{BESTBOUND} |, | \text{MIPOBJVAL} |)$$

It is also possible to set limits on the solve process, such as number of nodes (`MAXNODE` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MAXNODE.html>)), time limit (`MAXTIME` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MAXTIME.html>)) or on the number of solutions found (`MAXMIPSOL` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MAXMIPSOL.html>)). If the solve process is interrupted due to any of these limits, the problem will be left in its unfinished state. It is possible to resume the solve from an unfinished state by calling `XPRSmipoptimize` (`MIPOPTIMIZE`) again.

To return an unfinished problem to its starting state, where it can be modified again, the user should use the function `XPRSpstsolve` (`POSTSOLVE`). This function can be used to restore a problem from an interrupted global search even if the problem is not in a presolved state.

INTEGER PREPROCESSING

If `MIPPRESOLVE` (<https://www.fico.com/fico-xpress-optimization/docs/dms2019-03/solver/optimizer/HTML/MIPPRESOLVE.html>) has been set to a nonzero value before solving a MIP problem, integer preprocessing will be performed at each node of the branch and bound tree search (including the root node). This incorporates reduced cost tightening of bounds and tightening of implied variable bounds after branching. If a variable is fixed at a node, it remains fixed at all its child nodes, but it is not deleted from the matrix (unlike the variables fixed by presolve).

`MIPPRESOLVE` is a bitmap whose values are acted on as follows:

Bit	Value	Action
0	1	Reduced cost fixing;
1	2	Integer implication tightening.
2	4	<i>Unused</i>
3	8	Tightening of implied continuous variables.
4	16	Fixing of variables based on dual (i.e. optimality) implications.

So a value of $1+2=3$ for `MIPPRESOLVE` causes reduced cost fixing and tightening of implied bounds on integer variables.

or as otherwise set forth in a written software license agreement between you and FICO (or a FICO affiliate). Use of this documentation and the software described in it must conform strictly to the foregoing permitted uses, and no other use is permitted.