LocalSolver 3.0 に関する質問／回答　（2012.12.10）

LocalSolver 3.0 is released. You can download it for fun and profit! Relying on a unique technology based on pure and direct local search, LocalSolver 3.0 will provide better solutions faster to your largest mathematical optimization problems.

The main novelties in LocalSolver 3.0 are:

- Floating-point (double) coefficients are now allowed in any mathematical expression.

- Do not lose time to linearize your problem. Use our new highly-nonlinear mathematical operators: floor, ceil, log, exp, pow, cos, sin, tan, and many others. Note that you can use them in constraints and objectives.

  Please find the list of all mathematical operators supported by LocalSolver 3.0 in the Quick Start Guide at http://www.localsolver.com/mathematicalmodelingfeatures.html

- New local-search moves are available: specific moves for bin-packing constraints (that is, knapsack constraints coupled with assignment constraints), specific moves for sequencing & scheduling constraints, large neighborhood search thanks to compound moves.

  Sequencing or scheduling constraints are the ones arising in sequencing or scheduling problems. They are related to ordering some objects. For example, the car sequencing problem described at http://www.localsolver.com/exampletour.html?file=car_sequencing.zip, which consists in ordering vehicles on a assembly line, induce some sequencing constraints. All the scheduling problems consisting in assigning jobs to machines or men, with ordering or precedence constraints on jobs can be effectively tackled with LocalSolver 3.0. For example, this job shop scheduling problem :
  http://www.localsolver.com/exampletour.html?file=jobshop_scs.zip
  Compound moves are some local-search moves able to explore very large-scale neighborhoods. They correspond to a sequence of basic local-search moves. During the sequence of the basic moves, the solution may become infeasible (some constraints can be violated). This allows to escape hard local optima, and to reach better solutions. LocalSolver uses a large pool of moves which have different characteristics. We have some small, very time-efficient moves with low success probability, but we have also

some larger, less time-efficient moves but with a greater success probability. During the search, a "learning" module is in charge of choosing the best moves to apply, by making some statistics about the efficiency of the local-search moves. This allows us to reach both effectiveness (high-solution quality) and efficiency (short running time) on a large range of combinatorial problems

質問-1):

Learning module works only Step2. Is it correct?

.

回答)

The learning module is working in each stage to select online (that is, during the search) the best move to apply to optimize the current objective with the current constraints. So it is used in Step 1 (feasibility stage) and Step 2 (optimization stage). More details are given below.

- The learning algorithm for move selection (search component) is improved. It results clearly in faster convergence toward better solutions.

  The learning module allows to choose the best moves to apply during the search, that is on an instance solved with LocalSolver. It is not an off-line learning but an on-line automatic learning and tuning. More generally, it consists in adapting dynamically the behavior of LocalSolver dynamically (that is, during the resolution) in order to produce better solutions faster.

:

- The feasibility stage (that is, finding the first feasible solution) is faster and more robust thanks to a finer surrogate objective as well as local-search moves dedicated to this stage.

  The feasibility stage consists in finding a solution which respect all the constraints posed by the user in his model. To do that, all the constraints are first relaxed and LocalSolver optimizes a surrogate objective based on a measure of the infeasibility of each constraint. For example, for a constraint like "x + y + z == 1", the infeasibility measure is the gap between the value of the left member and the value of the right member, that is |x + y + z - 1|. All the measures are aggregated into one surrogate objective function to minimize. Some constraints are more important than other ones, in particular the

equality constraints (==), because generally harder to satisfy. When the surrogate objective is equal to 0, then a feasible solution is reached.
We apply some specific moves during the feasibility stage to optimize this surrogate infeasibility objective. In particular, the moves are targeted decisions involved in violated constraints in order to make them feasible. This allows to reach feasible solutions very quickly for problems which are not hardly constrained. We encourage users to avoid very hard constraints in their business optimization models, as explained in my previous email.

質問-2)

Is this below correct?

=Step1 (feasibility stage)

- Objective function is only the violations of constraints
- Search until the objective function becomes 0

=Step2 (optimization stage)

-Keeping the constraints, LocalSolver moves

=In the Step1 and 2, LocalSolver is designed to use SA.

回答)

This is correct. In Step 1 (feasibility stage), LocalSolver minimizes a surrogate objective based on the violation of constraints. Then, in Step 2 (optimization stage), LocalSolver optimizes the user objective, while maintaining the constraints feasible. In both cases, LocalSolver uses an adaptive simulated annealing heuristic with fast local search moves.

Some moves are common to the two stages, but some other ones are specific to the stage (feasibility or optimization). The learning module is working in each stage to select online (that is, during the search) the best move to apply to optimize the current objective with the current constraints.

In any case, we recommend to the users not to constraint too much their optimization model. The best way to proceed is to define very hard constraint (which are generally unlikely to be satisfied) as soft constraints. LocalSolver offers a nice feature for that: lexicographic-ordered multiple objectives. So you can minimize first the violations on some hard constraints as primary objective, and then maximize your revenue or minimize your costs as secondary objective. It is a better business optimization model (ensuring no "no solution found" message for your end-users) and a better mathematical optimization model for LocalSolver.

質問−3)

Regarding the expression of “some specific moves” above,
In here, LocalSolver is using Hyper Graph Theory,is it correct??
or another idea??

回答)

You can find some details on the local-search moves at http://www.localsolver.com/technology.html, and more particularly in our paper published in 2011 in the 4OR journal: http://pageperso.lif.univ-mrs.fr/~frederic.gardi/downloads/LocalSolver_4OR_2010.pdf.

Our local-search moves use the incidence hypergraph of constraints versus decisions. More particularly, the moves rely on the combinatorial structure and properties that this hypergraph induces. So in some sense, we are using Graph Theory in LocalSolver, but we use it at a very practical, algorithmic level. To be concrete, when we mention "scheduling moves", we say that we have designed moves which work effectively by relying on some properties that this hypergraph share for all scheduling problems.

Since this unique technique is at the heart of LocalSolver performance and superiority for tackling large-scale combinatorial problems, we cannot detail this one in deep details. We hope that the answer above is sufficiently detailed for your understanding.

- The capability to prove optimality or infeasibility is improved using combinatorial inference techniques.

  We apply some constraint propagation techniques to derived bounds on the objective function. This allows to detect infeasibility or to prove optimality in some cases. For example, on the car sequencing problem, we are able to compute good lower bound for the objective which has to be minimized. When the local-search is able to find solutions with cost equal to this lower bound, the search is stopped: this optimal is proved to be optimal. Note that the user can give his own lower bound (= his own objective or goal) to reach, to stop LocalSolver when a satisfactory solution is reached: this is done by using the command "setObjectiveBound()" in both LocalSolver modeler and programming APIs.
  I have attached a Word file which describes a case where LocalSolver is able to find and *prove* the optimal solution for car sequencing

(http://www.localsolver.com/exampletour.html?file=car_sequencing.zip) or steel mill slab design (http://www.localsolver.com/exampletour.html?file=steel_mill_slab_design.zip).

This makes LocalSolver better and faster on a very large spectrum of combinatorial optimization problems. On our benchmarks (100+ models), we improve all results of the previous 2.1 version. But more important, on the hardest problems, the gains both in solution quality and in running time are very large. For example, LocalSolver is now very effective on sequencing and scheduling problems.

Users having contracted maintenance with prior versions can download, install and use without extra cost LocalSolver 3.0: you just have to copy your previous license file into the 3.0 localsolver folder on your machine. For any information about this new 3.0 version, feel free to contact us.

Enjoy with LocalSolver 3.0!