Local Search for Mixed-Integer Nonlinear Optimization: a Methodology and an Application

Frédéric Gardi[1], Karim Nouioua[2]

[1] Bouygues e-lab, Paris, France

[2] Laboratoire d'Informatique Fondamentale – CNRS UMR 6166, Université Aix-Marseille II – Faculté des Sciences de Luminy, Marseille, France

fgardi@bouygues.com, karim.nouioua@lif.univ-mrs.fr

Abstract. A methodology is presented for tackling mixed-integer nonlinear optimization problems by local search, in particular large-scale real-life problems. This methodology is illustrated through the local-search heuristic implemented for solving an energy management problem posed by the EDF company in the context of the ROADEF/EURO Challenge 2010, an international competition of applied optimization. Our local-search approach is pure and direct: the problem is tackled frontally, without decomposition nor hybridization. In this way, both combinatorial and continuous decisions can be modified by a move during the search. Then, our work focuses on the diversification by the moves and on the performance of the incremental evaluation machinery. Exploring millions of feasible solutions within one hour of running time, the resulting local search allows us to obtain among the best results of the competition, in which 44 teams from 25 countries were engaged.

1 Presentation of the Problem

Électricité de France (EDF) is the historical French energy producer and supplier, whose operations and participations span worldwide today. The EDF power generation facilities in France stand for a total installed capacity of nearly 100 GW. Most of the French electricity is produced by thermal power plants: 90 % in 2009 among which 82 % by nuclear power plants. The subject of the ROADEF/EURO Challenge 2010, an international competition organized by the French Operational Research and Decision Support Society (ROADEF) and the Association of European Operational Research Societies (EURO), was focused on the medium-term (5 years) management of the EDF French thermal power park, and especially of nuclear plants which have to be repeatedly shut down for refueling and maintenance. Before describing our contributions (methodology and application), the main characteristics of this problem (decision variables, objectives, constraints) are outlined. For the sake of concision and readability, the presentation remains voluntarily informal. The interested reader is referred to the detailed technical specification provided by EDF in the context of the

ROADEF/EURO Challenge 2010 [9], which can be downloaded on the ROADEF website[3].

The thermal power park is composed of two kinds of plants. Type-1 (shortly T1) plants can be supplied in fuel continuously. They correspond to coal, fuel oil, or gas power facilities, or even virtual power stations allowing to import energy. On the other hand, Type-2 (shortly T2) plants have to be shut down for refueling and maintenance regularly. They correspond to nuclear power plants. Indeed, the fuel stock of these facilities is consumed as power is produced. Whenever a T2 plant is supplied with new fuel, it has to be offline and cannot produce during the length of this outage period, generally several weeks. Thus, the operation of a T2 plant is organized in a succession of cycles, namely an offline period (outage) followed by an online period (production campaign). These production assets are used to satisfy a customer demand over a specific time horizon. This horizon is discretized with a homogeneous time step. Customer load is uncertain and known only through an available set of uncertainty scenarios. These scenarios are assumed to be the realization of some stochastic processes (in particular weather conditions). Production at T1 plants incurs a cost proportional to the power output and also depends on the load scenario and the time step. For each T2 facility, the initial fuel stock at the beginning of the time horizon is known. Then, refueling of T2 plants leads to costs proportional to the amount of loaded fuel, also depending on the time step. Because refueling and maintenance are heavy operations immobilizing many resources, the order of cycles is fixed for each T2 power station. The earliest outages over the horizon cannot be canceled, and must be planned in given time intervals. For the latest outages, whose dates are not forced and which can be postponed, the following rule applies: if an outage is canceled, all following outages must be canceled too. Ultimately, the objective is to minimize the expected cost of production over the given horizon.

More precisely, the decision variables of the problem are: the starting dates of outages for all T2 plants, the refuel quantities for all outages of all T2 plants, the production levels for all T1 and T2 plants, all time steps, and all scenarios. Note that the first kind of decision variables are discrete, whereas the second and third ones are continuous. The objective function to minimize is composed of two terms: the expected production costs for all T1 plants (that is, the production costs over all scenarios divided by the number of scenarios), and the refuel costs for all T2 plants minus the expected stock values at the end of the period (to avoid end-of-side effects). The constraints can be classified into three categories; all these constraints are listed and numbered from CT1 to CT21 in the EDF specification. The first category (CT1) corresponds to constraints for coupling the production of the plants: for each scenario and each time step, the sum of productions for all T1 and T2 plants must be equal to the demand. The second category (CT2-CT12) concerns how plants can produce. The power of T1 plants must remain between minimum and maximum values depending on time steps and scenarios. When a T2 plant is offline, its power is equal to zero. When a T2 plant is online, its power must be non negative, lower than a maximum

[3] `http://challenge.roadef.org/2010/index.en.htm`

value depending on time steps. During the production campaign, the fuel level dynamics couples fuel quantity and production output over the time horizon: the fuel stock at time step $t + 1$ is equal to the fuel stock at t minus the energy produced at t, namely the product of the power delivered at t by the duration of a time step. However, the T2 power must respect an imposed profile when the fuel stock becomes lower than a given limit; this profile follows a decreasing piecewise affine function (with only a few pieces). If there is no longer enough fuel stock to produce, the power production is equal to zero. A nuclear plant produces generally at maximal power; otherwise, the T2 plant is said to be in modulation. Because the difference between the maximum power of the plant and the actual production leads to an undesirable wear of the equipment involved, the quantity of energy modulated (that is, not produced at maximal power) between two outages cannot be lower than a given value. For T2 plants, additional constraints are set on refueling operations. The refuel quantity must be in a given interval, the stocks before and after refueling must be lower than given limits. Note that at each refueling operation, a given percentage of stock is lost. Finally, the third category (CT13-CT21) corresponds to constraints on outage scheduling of T2 plants: earliest and latest starting dates of outages, minimum spacing or maximum overlapping of two outages, minimum spacing between the starting or ending dates of two outages, resource constraints (the number of maintenance teams able to perform refueling operations is limited), maximum number of outages containing a given week, maximum cumulated offline power during a given period.

This optimization problem can be classified as mixed-integer nonlinear. One can observe that it includes two dependent subproblems, nested according to a master/slave scheme. The master subproblem consists in determining a schedule of outages of T2 plants, satisfying constraints induced by limitations on resources which are necessary to perform refueling and maintenance operations (CT13-CT21). In summary, this subproblem involves combinatorial decision variables, subject to constraints related to intervals on the integer line. Having scheduling outages, the slave subproblem consists in planning production to satisfy the demand at the minimum cost, that is determining the stock refuels for each T2 plant and each outage, and the quantity of energy to produce by each plant (T1 and T2) at each time step for each scenario. This subproblem involves continuous decision variables subject to classical flow conservation and capacity constraints, but also to nonlinear constraints activated under certain logical conditions (CT1-CT12). The master subproblem of outage scheduling is theoretically NP-complete, because it corresponds to scheduling tasks with date and resource constraints [5]. On the other hand, the slave subproblem of production planning seems to be NP-hard too because of the nonlinear constraints (CT6, also called imposition constraints). Moreover, the instances to tackle may be very large: 8 outages to schedule over 300 weeks for 70 T2 plants, the production levels to determine for 170 plants (T1 and T2) over 10000 time steps for 500 scenarios. The execution time of the algorithm is limited to 1 hour on a standard computer.

We insist on the fact that the economic issues subjacent to this problem are considerable. The economic function to minimize (obfuscated in EDF data) contains in effect nine digits, representing nearly one billion euros per year of operating costs for EDF [6]. Therefore, a gap of only 0.1 % between two solutions represents savings of the order of one million euros a year.

2 Methodology and Outcomes

To the best of our knowledge, no work has been published yet in the literature which addresses the energy management model exposed above. Nevertheless, several softwares have been implemented by EDF researchers these last ten years for solving this problem. These solutions consists essentially in decomposing and reducing the problem so as to approach it with mixed-integer programming solvers [7, pp. 116–118]. The software currently exploited at EDF is based on a decomposition of the problem site by site, each subproblem being heuristically solved by integer linear programming according to the master/slave decomposition. Unfortunately, this solution does not guaranty the satisfiability of all the constraints of the problem. A large neighborhood search approach based on constraint programming (for scheduling outages) and integer linear programming (for planning production) was recently proposed by [8] (see also [7]), but the proposed solution does not take in input a set of scenarios but only one.

The solution approach that we have implemented in the context of the ROADEF/EURO Challenge 2010 is a randomized local search, technique which is rarely used in mixed-integer nonlinear optimization. The design and engineering of our local-search algorithm follow a precise methodology, inspired by the previous works of the authors [1–4]. This methodology is outlined below.

The first particularity of our local search is to be *pure* and *direct*. Indeed, *no decomposition* is done; the problem is tackled frontally. The search space explored by our algorithm is close to the original solution space. In particular, the combinatorial and continuous parts of the problem are treated together: combinatorial and continuous decisions can be simultaneously modified by a move during the search. By avoiding decompositions or reductions, no solution is lost and the probability to find good-quality ones is increased. Then, *no hybridization* is done: no particular metaheuristic is used, no tree-search technique is used. The diversification of the search is obtained by exploring a large variety of randomized neighborhoods. The second specificity of our local search is to be *very aggressive: millions of feasible solutions are visited within the time limit*. Indeed, randomized local search is a non deterministic, incomplete exploration of the search space. Therefore, exploring a huge number of (feasible) solutions during the allocated time augments the probability to find good-quality solutions.

Then, our local-search heuristic is composed of three layers: general heuristic, moves, evaluation machinery. The evaluation machinery forms the engine of the local search; it computes the impacts of moves on constraints and objectives during the search. The time spent to engineer each layer during the project follows the following distribution: 10 % on general heuristic, 20 % on moves, 70 % on

evaluation machinery. In summary, our work was focused on: *designing randomized moves allowing a diversified exploration* of the search space despite strong constraints, and *speeding up the evaluation of these moves* notably by implementing an *incremental randomized combinatorial algorithm* for solving approximately but very efficiently the continuous subproblem. Numerical experiments show that this combinatorial algorithm is *10 000 times faster* than state-of-the-art linear programming solvers (without imposition constraints), while providing near-optimal production plans. Note that the same approach was recently applied by one of the authors for solving a real-life inventory routing problem [1], which can be viewed as a mixed-integer linear optimization problem.

Benchmarks are divided into three categories A, B, X containing each one 5 instances. The instances A were communicated at the beginning of the qualification phase of the ROADEF/EURO Challenge 2010. Teams were selected for the final stage based on the results obtained on these instances. Then, instances B, much larger and very realistic, were given as test bed for the final stage. Ultimately, finalists were ranked according to their results on instances B (known from competitors) and instances X (unknown from competitors, communicated after the announcements of the final ranking). The final results were announced during EURO 2010, the 24th European Conference on Operational Research. Our algorithm was ranked 1st on instances A and B (among 44 teams engaged, 16 finalists), before falling to the 8th place due to a late-working-hours bug appearing on some instances X (note that only 4 teams among the 16 finalists have been able to provide all the solutions to instances X). Once corrected, our algorithm provides state-of-the-art results on instances X in conditions similar to ones of the Challenge. The results on instances B as computed by the Challenge's organizers[4] show an average gap greater than 1 % (resp. 10 %) between our solutions and the ones of the team ranked 3rd (resp. 6th). As evoked above, such gaps are important because corresponding from dozens to hundreds million euros of savings. One can observe that the majority of approaches proposed by the other competitors corresponds to MIP/CP-based decomposition heuristics.

The presentation of the local-search algorithm is done through three sections, each one corresponding to one layer of the local search. The last section is devoted to numerical experiments.

3 General Heuristic

The general heuristic is decomposed into three phases. First (phase 1), starting from a random scheduling of outages, we try to find an admissible scheduling of outages, that is, a scheduling of outages which respects all the combinatorial constraints of the problem (CT13-CT21). In this phase, fuel reloading and production levels are ignored. Then (phase 2), starting from the scheduling of outages previously found, we try to determine dates of outages in such a way that an admissible production plan exists. In particular, the spacing between

[4] `http://challenge.roadef.org/2010/finalResults.pdf`

outages of each plant must be sufficient so as to ensure that stocks do not exceed maximum levels before and after refueling operations (CT11). Once such a solution is found, an admissible solution of the original optimization problem is found. Finally (phase 3), this solution is optimized according to the original objective function.

Each phase is performed by local search. The heuristic employed in each phase is a *first-improvement descent with randomized selection of moves*; each phase follows the same design, allowing to factorize many components in their implementation. In practice, the first two phases are quickly passed (a few seconds). For the sake of efficiency, the phase 3 has been subdivided into several steps too. Indeed, the complexity for evaluating the cost of a solution depends on the number of scenarios. Since the number of scenarios may be large (up to 500), one way to reduce this complexity is to work on some subsets of scenarios (eventually aggregated). This is done as follows. At step s, the solution is optimized by local search for a subset $\mathcal{S}$ of scenarios, but with T1 completion costs computed over *all* scenarios (thanks to a special data structure). At step $s+1$, the previous solution is repaired so as to become feasible for the new subset $\mathcal{S}' \supset \mathcal{S}$ of scenarios, and then is optimized over $\mathcal{S}'$. Repairing a solution consists in adjusting the production of T2 plants in order to not to exceed the demand constraints (CT1) for the new scenarios; this is done by minimizing the amount of power exceeding the demand over all the time steps and for all scenarios. This sequence of steps can be parameterized in different ways to obtain the best ratio between efficiency and robustness, depending on the allowed time limit. After experiments on instances A and B provided by EDF, we have chosen to simply proceed as follows: first, we optimize on an "average demand" scenario, and then we refine the solution over all scenarios.

4 Moves

The combinatorial structure of the problem naturally induces the following move: select k outages in the current solution and shift them over the time line. The neighborhood induced by such a move is of size $O(H^k)$, with H the number of weeks given in input. In the preliminary phase of the Challenge, these moves were applied totally randomly by the heuristic with small values of k (between 1 and 3) in a first-improvement fashion (if the new solution reached via the move satisfy all the constraints and has a better cost, then commit the move, otherwise rollback it). Unfortunately, the combinatorial part of the problem being hardly constrained, such "simple" moves have a low success rate, which limits the diversification and may prevent the convergence to high-quality solutions.

To overcome this difficulty, larger moves have to be designed so as to better explore the combinatorial part of the problem. This has been done by designing *compound moves* based on the simple moves described above. The underlying principal of these compound moves is to allow to *join (better) admissible solutions by passing through infeasible solutions*. It is particularly useful when the constraints of the problem are hard, as it is the case here. A compound move

is designed as follows. First, apply one move which may destroy the feasibility of the current solution. Then, apply iteratively some moves in order to repair the solution. In fact, repairing the solution using simple moves is equivalent to applying local search for solving a satisfaction problem. In this way, an appropriate objective to optimize must be defined to guide local search toward feasible solutions. It can be simply done by defining an objective based on the number and the amplitude of violations for each type of constraints. These compound moves can be viewed as a generalization (and simplification) of ejection chains and destroy-repair methods (see [10] for more details on these search techniques).

In order to speed up the convergence toward feasible solutions, the "repairing" moves, applied following a first-improvement descent, target outages inducing violations. Moreover, the number of simple moves to attempt during reparation is limited: a compromise must be found between the time spent in attempting the compound move and its success rate (both depending on its length). On the other hand, in order to refine the search, the "destroying" move is chosen randomly (following a non uniform distribution) in a pool composed of the following moves:

- `k-MoveOutagesRandom`: select k outages among T2 plants randomly and move them randomly in "feasible" time intervals, that is, ensuring the respect of earliest and latest starting dates (CT13 constraints) and maximum stocks before and after refueling (CT11 constraints);
- `k-MoveOutagesConstrained`: select T2 plants which are involved in combinatorial constraints related to outage scheduling (CT14-CT21) randomly, select k outages of these plants randomly, move these outages randomly in feasible time intervals;
- `k-MoveOutagesConsecutive`: select a T2 plant randomly, select k consecutive outages of these plant randomly, move these outages randomly in feasible time intervals.

Note that more than the half of destroying moves attempted are of type `1-MoveOutagesRandom`, but specific and larger moves helps to diversify the search. Thus, whereas the admissibility rate of simple moves (that is, the number of moves leading to a new feasible solution, divided by the number of attempted moves) is about 20 % on average, the admissibility rate of compound moves is about 75 %. The computational experiments presented in last section show that a first-improvement descent with randomized compound moves converges confidently toward high-quality solutions (no local optimum is encountered after hours of computations).

5 Evaluation Machinery

As suggested in introduction, the effectiveness of an incomplete search depends crucially on the number of (feasible) solutions explored. Hence, an important task arising in engineering a local-search approach is to make the evaluation fast, especially when tackling such large-scale real-life problems. Methodologically, our

work on this point is driven by a simple goal: to lower as much as possible the *practical* time complexity of the evaluation of all moves. For this, an incremental evaluation is necessary, exploiting the invariants between the current solution and the one induced by the move. In the present case, the evaluation of the compound move is realized in two steps, each one corresponding to the resolution of a hard subproblem: first (re)scheduling outages, then (re)planning production (refuels and levels). Here is the general scheme of the evaluation layer, for a set S' of scenarios:

Combinatorial part:
  perform a destroying move;
  while combinatorial violations remain **do**
    perform a repairing move;
Continuous part:
  set refueling amounts of impacted outages;
  for each scenario in S' **do**
    set production levels of impacted T2 plants;
  compute global cost of new solution;

5.1 Combinatorial Part

When a simple move is applied for satisfying combinatorial constraints (CT14-CT21), the violations on these constraints are maintained incrementally through routines related to the arithmetic of integer intervals (distance, intersection, inclusion). For each constraint, the evaluation returns not only if the constraint is violated or not following the move, but also the "distance" to feasibility (which can be interpreted as the cost of infeasibility). For example, for a minimum spacing constraint of 10 weeks between two outages, this cost shall be 8 if the spacing is of 2 weeks after the move. For constraints concerning minimum spacing/maximum overlapping between outages (CT14-CT18), the evaluation for k' outages impacted by the move is done in $O(k'k)$ time with k the number of outages involved with the constraint. For resource constraints (CT19), the evaluation for k' impacted outages takes $O(\sum_{k'} w_i)$ time with w_i the number of weeks for which a resource is immobilized during outage i. For constraints bounding the number of overlapping outages during a given week (CT20), the evaluation for k' impacted outages is done in $O(k')$ time. For constraints limiting the offline power capacity of T2 plants during a time period (CT21), the evaluation for k' impacted outages takes $O(\sum_{k'} t_i)$ time with t_i the number of time steps during outage i. Note that in practice our moves are such that the number k' of impacted outages is small relatively to the total number of outages scheduled ($k' = O(1)$). As explained in the previous section, the convergence toward combinatorial feasible solutions is speeded up by attempting moves on outages inducing violations. The randomized selection of outages inducing violations is made in constant time by maintaining a bipartition of outages, with the ones inducing violations on the left side and the others on the right side, after each evaluation of combinatorial constraints.

In addition to CT14-CT21 combinatorial constraints, we add an implicit constraint, called "cut of minimum distance between outages", induced by the continuous subproblem: the distance between two consecutive outages k and $k+1$ must be large enough to ensure that stocks at outage $k+1$ do not exceed maximum levels before and after refueling operations (CT11), even if the fuel reload at outage k is *minimal* and the production during cycle k is *maximal*. When an outage is moved, this minimal distance can be reevaluated in $O(\log d)$ worst-case time, with d the number of time steps between outages. For each T2 plant, compute the cumulated maximum powers of the plant from the beginning to the end of the horizon, for all time steps. Using this data structure, the maximum amount of fuel which can be consumed between any pair $t1 \leq t2$ of time steps is obtained in constant time. Then, starting a production cycle with the minimum fuel level (derived from the minimum fuel reload), the resulting fuel level after t time steps of production at maximum power is computed in constant time. Therefore, the minimum number of time steps such that the resulting fuel level does not exceed a given level (CT11) is obtained by dichotomy in $O(\log d)$ time in the worst case. In practice, this evaluation is still made faster by caching the output minimum distance with the key (starting time step, starting fuel level) in a hash map. Ultimately, it allows to compute the desired minimum distance in *amortized constant time*. Since 80 % of the evaluation time is spent in the continuous part, cutting the evaluation earlier using this property of minimum distance between outages is crucial for the efficiency of the whole local search.

5.2 Continuous Part

Now, assume that the compound move yields a solution which is combinatorially feasible (CT13-CT21 constraints + cuts of minimum distance between outages). The continuous decision variables impacted by the move must be updated so as to ensure the feasibility on the continuous part; then, the global cost of the new solution induced by the move must be evaluated. This continuous subproblem, whose objective is (roughly speaking) to minimize T1 completion costs, is solved by an *incremental randomized combinatorial algorithm*.

For each impacted T2 plant, the refuel at outage k is determined randomly between the minimum given in input (CT7) and a maximum reinforced to avoid exceeding the maximum levels before and after refueling operations at outage $k+1$ (CT11). The computation of this maximum value is based on the maximal consumable energy $e_{\max}(s)$ over the production cycle and the resulting minimal ending fuel level $l_{\min}(s)$ for each scenario s. These values are computed for each scenario through a first pass over the time steps from outage k to outage $k+1$. A nuclear plant produces generally at maximal power; otherwise, the plant is said to modulate. Modulation is sometimes necessary when the demand is too low compared to the T2 power capacities. But it can also be used to reduce production costs by transferring T2 power from time steps where T1 costs are cheap to time steps where T1 costs are expensive. The quantity of energy which can be modulated (that is, not produced at maximal power) for each scenario, denoted by $e_{\text{mod}}(s)$, is obtained as the minimum among the maximum bound

given in input (CT12) and the maximum stock before (resp. after) refuel at $k+1$ minus the minimal ending fuel stock $l_{\min}(s)$. The energy $e(s)$ to consume during the cycle is then determined randomly between $e_{\max}(s) - e_{\mathrm{mod}}(s)$ and $e_{\max}(s)$. Finally, through another pass, the power levels between outages k and $k+1$ are set so as to consume $e(s)$. This is done randomly from the left to the right or guided by the lowest T1 completion costs. In summary, the randomized algorithm for planning production is running in $O(T')$ time for each scenario, with T' the number of time steps between outages k and $k+1$. Thus, the refuels and the production levels which are impacted by the move are recomputed in $O(P_2'TS')$ time, with P_2' the number of impacted T2 plants, T the number of time steps over the planning horizon, and S' the number of considered scenarios.

Finally, the global cost of the new solution is evaluated. The most time-consuming task is the evaluation of the T1 completion costs (when T2 plants cannot provide enough power to satisfy demands). For each impacted T2 plant and each time step, the cheapest T1 completion can be computed in $\log(P_1S')$ time, with P_1 the number of T1 plants and S' the number of considered scenarios. For a given scenario, one can observe that the cheapest completion cost can be computed by dichotomy in $\log P_1$ time, using a data structure containing the powers and costs of T1 plants, cumulated according to nondecreasing costs. By fusioning the S' structures into only one, it is possible to perform the computation in $\log(P_1S')$ time for S' scenarios.

In conclusion, the time complexity for evaluating the continuous part (refuels, production levels, global cost) is *almost linear in the number of modifications induced by the move* on the current solution. This time complexity is critical in the global performance of the local search. Having relaxed imposition constraints, the continuous subproblem can be solved by linear programming. Some numerical experiments have shown that our combinatorial approximation algorithm is *10 000 times faster* than the state-of-the-art linear programming solvers (Gurobi Optimizer and IBM ILOG CPLEX), while providing production plans with optimality gap lower than 0.1 %.

6 Numerical Experiments

The whole algorithm was implemented in ISO C++ programming language (C++0x). The resulting program includes nearly 12 000 lines of code, whose 15 % are dedicated to check the validity of all incremental data structures at each iteration (only active in debug mode). Note that the continuous part of the problem was handled with exact precision using 64-bits integers. The executable was statically compiled on Linux x64 platform using GCC 4 with full optimization (-O3). Gprof and Valgrind were used for CPU and memory profiling.

All statistics and results presented here have been obtained (without parallelization) on a computer equipped with a Linux x64 operating system and a chipset Intel Xeon X5365 (CPU 3 GHz, RAM 4 GB, L2 4 MB, L1 64 kB). Benchmarks are divided into three categories A, B, X containing each one 5 instances. Note that the scale of instances B and X is much larger than instances A. The

running time limit was fixed to 1 hour by the organizers of the Challenge. Note that 1.7 GB of RAM are allocated for tackling the largest instances (B8, B9, B10, X13, X14, X15). Since our local-search heuristic is randomized, the results which are given correspond to averages over 10 runs with different seeds. The variance of results is weak (lower than 0.1 %), except for instances B8 and B9 (0.5 %). All the solutions obtained passed the EDF's checker.

Table 1. Results of our local-search heuristic for 10 minutes, 1 hour, and 10 hours of running time, and comparison with the best results found by the other competitors of the Challenge (1 hour).

instance	10 mn	1 h	10 h	best	gap
A01	1.694 804 e11	1.694 780 e11	1.694 748 e11	1.695 383 e11	−0.036 %
A02	1.459 699 e11	1.459 600 e11	1.459 568 e11	1.460 484 e11	−0.061 %
A03	1.543 227 e11	1.543 212 e11	1.543 160 e11	1.544 298 e11	−0.070 %
A04	1.115 163 e11	1.114 966 e11	1.114 940 e11	1.115 913 e11	−0.085 %
A05	1.245 784 e11	1.245 599 e11	1.245 439 e11	1.258 222 e11	−0.989 %
B06	8.413 041 e10	8.387 786 e10	8.379 878 e10	8.342 471 e10	+0.543 %
B07	8.118 554 e10	8.117 563 e10	8.109 972 e10	8.129 041 e10	−0.129 %
B08	8.231 156 e10	8.196 477 e10	8.189 974 e10	8.192 620 e10	+0.047 %
B09	8.219 982 e10	8.175 367 e10	8.168 956 e10	8.261 495 e10	−1.043 %
B10	7.805 363 e10	7.803 998 e10	7.791 096 e10	7.776 702 e10	+0.351 %
X11	7.919 385 e10	7.910 063 e10	7.900 765 e10	7.911 677 e10	−0.020 %
X12	7.760 939 e10	7.760 090 e10	7.756 399 e10	7.763 413 e10	−0.043 %
X13	7.652 986 e10	7.637 339 e10	7.628 852 e10	7.644 920 e10	−0.099 %
X14	7.631 402 e10	7.615 823 e10	7.614 948 e10	7.617 299 e10	−0.019 %
X15	7.444 765 e10	7.439 302 e10	7.438 837 e10	7.510 139 e10	−0.943 %

Table 1 contains the results obtained for each instance with 10 minutes, 1 hour, and 10 hours of running time respectively. The convergence of the local search is very fast: on average, 99.9 % of the improvement is performed in less than 10 minutes. On average, our algorithm attempts more than 20 000 compound moves per minute on largest instances (which corresponds to more than 100 000 attempted simple moves); 75 % of these compound moves lead to new feasible solutions. The success rate of simple moves during the compound move (to improve the number of violations on combinatorial constraints during the reparation) is greater than 20 %, while the success rate of compound move (to improve the cost of the current solution) is nearly 1 %. Consequently, *the number of admissible solutions visited within 1 hour is at least one million*, while the number of improving solutions is of a few thousands. One can observe that despite using a standard descent heuristic, diversified compound moves allows a large diversification, which allows still finding improving solutions after several hours of running time.

In Table 1, a comparison is done between the solutions found by our local-search algorithm and the best solutions found among the competitors within 1 hour of running time (computed by the Challenge's organizers). The latter ones

have been obtained on a comparable platform: Linux x64 operating system with chipset Intel Xeon 5420 (2.67 GHz, RAM 8 GB, L2 6 MB, L1 64 kB). Over the 15 instances, we obtain 12 best solutions. In fact, the solutions provided by the team Kuipers-Peekstok, also obtained with a direct local-search heuristic, are very close to ours (average gap lower than 0.1 %). But the average gaps of solution approaches ranked after the 3th (resp. 6th) position, mainly based on MIP/CP decompositions, is greater than 1 % (resp. 10 %). Such gaps are considerable here since the economic fonction (obfuscated in EDF data) contains in effect 9 digits [6].

Acknowledgements. We warmly thank Dr. Bertrand Estellon (LIF Marseille, France), which worked on several topics in relation with this paper and helped us during the Challenge to perform some numerical experiments.

References

1. Benoist, T., Estellon, B., Gardi, F., Jeanjean, A.: Randomized local search for real-life inventory routing. Transportation Science (2011), to appear
2. Estellon, B., Gardi, F., Nouioua, K.: Large neighborhood improvements for solving car sequencing problems. RAIRO Operations Research 40(4), 355–379 (2006)
3. Estellon, B., Gardi, F., Nouioua, K.: Two local search approaches for solving real-life car sequencing problems. European Journal of Operational Research 191(3), 928–944 (2008)
4. Estellon, B., Gardi, F., Nouioua, K.: High-performance local search for task scheduling with human resource allocation. In: T. Stützle, M. Birattari, H.H. (ed.) SLS 2009, the 2nd International Workshop on Engineering Stochastic Local Search Algorithms. Lecture Notes in Computer Science, vol. 5752, pp. 1–15. Springer, Berlin, Germany (2009)
5. Garey, M., Johnson, D. (eds.): Computers and Intractability: a Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Science, W.H. Freeman and Companny, New-York, NY, USA (1979)
6. Gorge, A.: Planification des arrêts pour rechargement des centrales nucléaires. JOR 2008, la 3ème Journée de Recherche Opérationnelle et Optimisation dans les Réseaux, Paris, France (May 2008), oral communication
7. Khemmoudj, M.: Modélisation et résolution de systèmes de contraintes : application au problème de placement des arrêts et de la production des réacteurs nucléaires d'EDF. Ph.D. thesis, Université Paris 13 (Paris-Nord), France (2007)
8. Khemmoudj, M., Porcheron, M., Bennaceur, H.: When constraint programming and local search solve the scheduling problem of Électricité de france nuclear power plant outages. In: Benhamou, F. (ed.) CP 2006, the 12th International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 4204, pp. 271–283. Springer, Berlin, Germany (2006)
9. Porcheron, M., Gorge, A., Juan, O., Simovic, T., Dereu, G.: Challenge ROADEF/EURO 2010: a large-scale energy management problem with varied constraints. EDF R&D, Clamart, France (February 2010), 27 pages
10. Rego, C., Glover, F.: Local search and metaheuristics. In: Gutin, G., Punnen, A. (eds.) The Traveling Salesman Problem and Its Variations, pp. 105–109. SIAM Monographs on Discrete Mathematics and Applications 9, Kluwer Academic Publishers, Dordrecht, Netherlands (2002)