Path Relinking Based Intensification Strategies for a Simulation-Optimization Scheduling Problem Arising in Hydroinformatics

Maddalena Nonato and Andrea Peano (✉)

Engineering Department - University of Ferrara
Via Saragat 1 - Ferrara - 44121 - Italy
{maddalena.nonato,andrea.peano}@unife.it

Abstract. Water contamination events in a urban hydraulic network can be handled by operating two network devices: valves, to divert flows, and hydrants, to dispel flows. No remote control can activate these devices, which must be operated on site, therefore teams of technicians have to move from device to device on the city streets. The response to the contamination corresponds to a feasible schedule of the devices activation time, which can be modelled as a multiple Travelling Salesman Problem; the quality of the schedule is the volume of contaminated water consumed by the users untill water quality returns to safety. This quantity is highly sensitive to the schedule and is computed by way of a computational demanding hydraulic simulation, which takes about 5 seconds for the Ferrara's hydraulic network serving around 130'000 citizens. Previous studies proved that minimizing the makespan, as it would be intuitive in case of emergency, yields worse solutions than random approaches. Genetic Algorithms were proposed to optimize the schedules, however they converge to local optima, depending on the initial population. We propose to apply Path Relinking to explore the space between pairs of local optima; the experiments showed that this technique improves the local best. This work is a follow up of what presented at AIxIA 2015, enpowered with a light introduction to Path Relinking and its applications. Moreover, an extended experimental campaign has been carried out on all the 42 contamination scenarios.

Keywords: scheduling; path relinking; simulation-optimization

1 Introduction

Hydroinformatics is a new, promising, interdisciplinary research field arising at the junction of Hydraulic Engineering and Computer Science, in which complex decision problems related to water management applications are modelled and solved by way of quantitative solution tools developed within well assessed computer science methodological paradigms, such as Constrained Programming on Finite Domain and Mathematical Programming Optimization. Several such examples can be found in the literature which exploit the network based problem structure, taking advantage of solution methodologies already developed for

transportation and communication networks, as earlier pointed out by Simonis in a seminal work [34]. Among the most recent contributions, let us mention: the design of the expansion of a Water Distribution System (WDS) combining global search techniques with local search [5], the optimal location on the WDS of water quality sensors in order to early detect water contamination exploiting integer programming based location models [26] and objective function submodularity [22], the optimal scheduling of devices controlling field irrigation [19] to meet farmers irrigation time demands, the optimal location of isolation valves on the pipes of a WDS to minimize service disruption in case of maintenance operations [7] and [9], and the scheduling of devices activation as a countermeasure to contamination events [10], which is the problem we deal with in this paper.

In all cases, the feasible solutions have to meet complex technological requirements which are modelled by the constraints of the optimization problem, while the objective function describes how the hydraulic system reacts to certain values of the parameters, which are the model variables. Quite often, the system reaction can not be encoded by analytical closed formulas but it is the result of a computational demanding simulation process, which poses a challenge to the development of solution methodologies able to scale efficiently and tackle real life instances.

In this paper we deal with the last mentioned problem, namely computing the optimal activation time of a set of devices located on the WDS. In case of a contamination event, the devices activation times alter water flow, influence how contaminant spreads in the network, and determine at which concentration contaminant reaches demand nodes where drinking water is consumed by system users. An optimal schedule is a set of activation times which minimizes the volume of consumed contaminated water. A feasible schedule is a set of activation times according to which the teams of technicians, due to manually activate the devices on site, can reach the selected device on time, travelling on the street network when moving from a device to the next. Common practice at most water distribution agencies is to compute such schedules by hand, driven by common sense principles, such as "the sooner the better". Recent approaches exploited genetic algorithms (GA) to compute better schedules automatically [2, 10, 17, 28]. In this paper, we extend the study presented at AIxIA 2015 [27] devoted to investigate how hybrid solution approaches for such complicated real problems can largely benefit from the integration of different search paradigms.

We also believe that path relinking, owning to its applicability to basically any combinatorial optimization problem, could be as well integrated within most CP engines, acting as an intensification module to be activated on any set of good quality (partial) solutions visited so far in the search tree.

In the rest of the paper, first we introduce the problem and recall the solution strategy based on Mixed Integer Linear Programmig (MILP) hybridized genetic algorithms developed so far, pointing out at some deficiencies. Then, we describe a neighbourhood based search strategy, so called *Path Relinking* (PR) originally proposed by Glover [15], which intensifies the search within a section of the feasible region to which a set of high quality solutions belong. We present how to

use PR in our problem, compare two different neighbourhood structures to build the search path connecting two solutions, and asses the efficacy of the approach by experimental results computed on real data for the WDS of a medium size city in Italy, showing how by enhancing our GA with a post optimization PR phase can improve the approach robustness and partially mend the present flaws.

2 Problem Description

WDSs are essential components of our daily life as they bring clean, safe drinking water to customers every day. At the same time, WDSs are among the most vulnerable infrastructures, highly exposed to the risk of contamination by chemical and biological agents, either accidental or intentional. A WDS is a complex arrangement of interconnected pipes, pumps, tanks, hydrants and valves, whose large planimetric extent (a small city network may reach $200km$ and a thousand of pipes and nodes) and sparse topology make full surveillance impossible. Therefore monitoring is the only viable alternative. In practice, a set of water quality sensors is located at different sites on the WDS to test water safety in real time, looking for the presence of potential contaminants [26]. Their location is strategically determined so that a contamination event is detected as soon as possible, based on a set of contamination scenarios.

Contaminant quickly spreads through the network and population alerting strategies may not entirely ward off users' water consumption. When the network is fully districted, the sector where the alarm is raised can be seamlessly disconnected from the rest of the network, but this is rarely the case. In general, despite of the hazard, water supply can not be completely cut off. The shut down of the entire system would disrupt those security related functions that rely on continuous water supply, such as fire police service or water based cooling systems at large, production intensive, industrial facilities. Therefore, beside population warning procedures, countermeasures devoted to divert the contaminant flow away from high demand concentration sectors must be set up, aiming at mitigating population harm.

An effective way of altering water flow is by activating some of the devices which are part of the system, namely by closing isolation valves and opening hydrants, in order to achieve contaminant isolation, containment, and flushing. In particular, opening hydrants can expel contaminated water, while contaminated pipes can be isolated by closing their isolation valves. Due to the highly non linear functional dependencies that link water flow and the time at which a given device is operated, the global effect of a schedule, i.e., the vector of activation times for the selected devices, can not be decomposed into the sum of the effects of the activation of each individual device, nor the effect of a local change in the schedule can be anticipated. On the contrary, the only way to evaluate the volume of consumed contaminated water due to a schedule is by a computationally intensive simulation. In other words, we are optimizing a black box function and solving a so called simulation-optimization problem [3]. The chosen simulation package is EPANET [32], a discrete event-based simulator which represents the

state of the art in Hydraulic Engineering. EPANET is given a schedule, the description of the hydraulic network, the expected water demand over time, and a contamination scenario, and it yields the volume of contaminated consumed water (we speak of contamination when concentration level is above the danger threshold of $0.3mg/ml$ that causes human death if ingested). Simulation is computationally intensive and it is the bottleneck of any solution approach.

In our case study, each simulation call takes on average $5''$, which poses a limit on the number of function evaluation calls, despite of the fact that the problem is solved offline; this fact influences the choice of the search strategy, as discussed in [11]. Moreover, there is no a priori information on what a good schedule should look like, and common sense inspired criteria such as "the sooner the better" lead to low quality solutions. In conclusion, there is no way to distinguish between a good and a bad schedule without simulation.
So far, it concerns the objective function of our simulation optimization problem. Regarding the solution feasibility, a schedule $t^{\mathcal{F}}$ is feasible provided that there is an assignment of the n devices to the m teams available and, for each team, its devices can be sequenced so that if device j follows device i the difference between the respective activation times in the schedule is equal to τ_{ij}, i.e., the travelling time on the street network from the location of device i to the location of device j. All teams gather at the mobilization point at a given time after the alarm is raised (according to the protocol) and conventionally the departure time is set to 0. This maps the feasible region of our problem into the one of a well known combinatorial optimization problem, the *open m-Travelling Salesman Problem* [4] (mTSP). This providing a graph representation of our problem where the mobilization point is the depot, each device is a client node of the graph and each team visits the assigned devices travelling along a route starting from the depot and ending at its last visited client.

All these features lead us to set up a genetic algorithm hybridized with a Mixed Integer Linear Programming solver which encapsulates the feasibility constraints within the cross over operators, as described in detail in [11]. In that paper, several computational experiments were carried out to calibrate population size and number of generations for a single GA run. Moreover, we verified the poor quality of the solutions obtained according to heuristic criteria, such as minimum makespan or minimizing the sum of the activation times. Besides, neighbourhood based local search were tested and proved to be not competitive given the limited number of solution evaluations allowed, since the search trajectory remains confined not far from the starting point. On the contrary, the literature confirms that in such cases population based heuristics, which carry on a broader search and are able to explore a wider part of the feasible region, are able to provide better results [3].

Although we could improve by far and large the best solutions available in the literature for our case study, that solution approach has a typical GA flaw, that is, it converges to a local optimum which depends on the starting population. However, the differences among different local optima, obtained by repeated runs starting from different initial populations, vary depending on the contamination

scenario. Since the number of function evaluations is limited, in this study we address the question concerning what is the best way of exploiting the computational resources we are allowed, and if there is a way to take advantage of the knowledge of a set of different, high quality solutions.

In the next section we first provide a light introduction to path relinking and then we describe how PR can provide a pattern search that fulfils these expectations.

3 Intensification by path relinking

As mentioned, in this application GAs often yield high quality solutions that depend on the initial population: this is due to the existence of several local optima. These different solutions identify a promising subregion of the feasible space, which is worth further inspection, according to some exploration strategy. Classical Local Search transforms a solution gradually: at each step it moves from a solution to an improving one in the current neighbourhood, driven by the objective function. In our case, the search goes from one local optimum to another, by gradually making the current solution more similar to the final one. This search is not guided by the objective function, but rather by a *distance* criterion, and quite often a better solution is found along this search trajectory. This philosophy lies at the heart of an intensification technique named PR [15].

Working on a *reference set* (rs) composed of several solutions, PR first selects from rs an *initial* reference (r) and a *guiding* (often called *target*) (g) solution, then it iterates valid moves to transform step by step r into g. Figure 1 shows a graphical representation of the transformation of r into g, differing initially on 4 elements; so 3 intermediate solutions are selected, namely r^1, r^2, and r^3. This procedure allows for the exploration of the path between two good solutions, according to the hypothesis that a better one can be found among the feasible solutions in the middle.

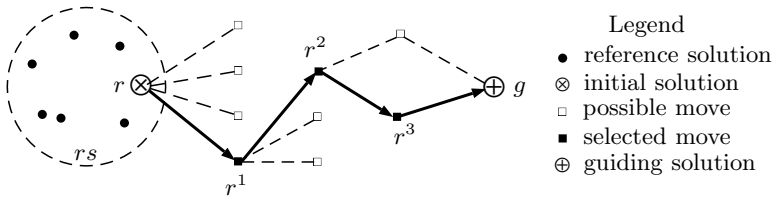

Fig. 1. Graphical representation of path relinking

Since PR builds a new solution starting from the features of two elite solutions, it can be also seen as an evolutionary algorithm, in which randomness is substituted by a deterministic search strategy that draws the possible path between two feasible solutions.

The building blocks of a path relinking algorithm are:

- the reference set and its construction;
- the reference and the target solutions and their selection;
- the path between two solutions, i.e., the neighbourhood structure.

Several variants and generalizations of PR are possible, which are elegantly discussed in [16], such as truncating the search on a path to resume it on another (either new or existing) path of the same $g - r$ couple, or different policies for choosing the move in the current neighbourhood rather than moving on the best one. In this work we adopt this last classical strategy since we prefer to spend the limited number of solution evaluations to explore the "best" path between different $r - g$ pairs rather than several paths between the same $r - g$ pair.

In our case, the bunch of best populations given by some GA runs provides naturally the dataset the reference set can be built up from, the target g can be easily selected as the best solution in rs; and the reference r has to be selected properly through quality and diversity criteria, as Section 3.1 reports.

3.1 Selection of reference candidates

As stated before, the reference set can be built up from the final populations of the GAs. In particular, diversity from the target beside quality should be taken into account, since the number of inspected solutions grows with distance to target; thus, different metrics can be combined together to filter properly the initial dataset and provide a set of good quality solutions not too close to each other.

The distance between two solutions can be evaluated considering the routes of the teams as well as the activation times. Despite in the former studies the diversity is measured on the graph representation of the routing problems [20, 29–31, 35], in this case the preferred way is to measure the diversity over the time representation. In fact the graph representations of the solution would introduce a huge amount of redundancy [6], this means that the same vector of activation times can be mapped into different trees that may differ a lot with respect to the metrics defined for graph representations. On the opposite, in this problem, what makes a solution different from another one is the difference between the associated schedule. There are a few choices to measure the difference between two vectors; in particular, the metrics here proposed for the time representation are the Hamming distance

$$h(g, r) = \sum_{i=1}^{N_{dev}} d_i \tag{1}$$

where g_i (r_i) is the activation time of device i in solution g (r) and $d_i = 1$ if $g_i \neq r_i$ while $d_i = 0$ otherwise, and the euclidean distance

$$e(g, r) = \sqrt{\sum_{i=1}^{N_{dev}} (g_i - r_i)^2} \tag{2}$$

between two vectors of activation times g and r. The former gives a measure about how many elements differ in the vectors, whereas the latter measures how much the vectors differ in terms of activation times. In order to prevent the inclusion of too similar vectors in rs, two thresholds β and γ are defined: given the target g a solution r is included only if $h(g,r) \geq \beta$ and $e(g,r) \geq \gamma$.

As far as the quality of the reference set, a proper metric is to consider only solutions having quality within a certain percentage distance δ from the target's one, i.e., $\frac{q(r)-q(g)}{q(g)} \leq \delta$ holds for any $r \in rs$.

Finally, the choice of β, γ, and δ is really important, even more whenever the number of evaluations is limited. In fact, in this case excluding a promising solution may affect hugely the effectiveness of the approach.

3.2 A Path Relinking Based on Sequences

Building on the structure of the feasibility constraints, that is the one of the mTSP, path relinking for routing problem in general can provide useful suggestions [35]. Indeed, path relinking for routing problems works on symbolic representations of routes, in which any route is expressed by an ordered set of visited customers [20, 35, 30]. For 3 teams v_1,v_2, and v_3, and 7 customers, namely $c_1, \ldots, c_7$, a feasible solution assigns a route to each vehicle, e.g., $v_1 = \{c_1, c_2\}$, $v_2 = \{c_3, c_4\}$, $v_3 = \{c_5, c_6, c_7\}$. Equal solutions visit the customers along same routes. To transform a solution into a different one, every customer should be *relocated* into the right position of the right route. In path relinking for routing problems, this is done iteratively by relocating one customer at each step.

The same representation can be used for mTSP. Since the routes have no names, the order of the devices within the routes is the only valuable information; moreover, in a route any device follows either the depot or a unique device, thus the order of activation within the routes can be stated by listing the devices' *predecessors*, i.e., a list of tuples "(h_p, h_s)" meaning that device h_p precedes device h_s in the solution. For example, in the solutions in Figure 2, three teams work overall on seven hydraulic devices, namely $1, \ldots, 7$; the *initial* solution r and the guiding solution g can be represented by the following predecessor lists:

$$P_r = \{(d,3),(d,4),(d,6),(1,5),(4,7),(6,1),(7,2)\},$$

$$P_g = \{(d,3),(d,4),(d,6),(1,7),(3,5),(4,1),(5,2)\}.$$

In general, two solutions a and b are equal iff $P_a = P_b$; whereas, whenever two solutions differ, the predecessors of a that are not in b are $P_{a-b} = P_a \setminus P_a \cap P_b$, vice versa for b is $P_{b-a} = P_b \setminus P_a \cap P_b$. Moreover, the cardinality $card(P_{a-b})$ measures the distance of a from b, with $card(P_{a-b}) = card(P_{b-a})$. For example, for r and g we have that $P_{r-g} = \{(1,5),(4,7),(6,1),(7,2)\}$, and $P_{g-r} = \{(1,7),(3,5),(4,1),(5,2)\}$; so, the distance between r and g is 4.

To get closer to b starting from the configuration of a, at least one device h_s such that $(h_p, h_s) \in P_{b-a}$ should be relocated after its predecessor h_p in b; in this sense, the set P_{b-a} contains the possible moves to transform a into

b. For example, $(5, 2) \in P_{g-r}$ means that device 2 needs to be relocated right after 5, making r more similar and one step closer to g. This means that the neighbourhood of r with respect to g is the set of solutions $\mathcal{N}(r, g) = \{r_n \mid card(P_{g-r} \setminus P_{g-r_n}) = 1\}$. In other words, the neighbourhood of r with respect to g is the set of solutions r_n obtained by relocating one device in r according with P_{g-r}.

At the k^{th} iteration, PR has to choose which device $h_s : (h_s, h_p) \in P^k_{g-r}$ is relocated, in order to move to r^{k+1}. To make this choice it evaluates by EPANET every $r^k_n \in \mathcal{N}^k$, and moves to the one with the lowest volume; this solution becomes the reference of the next step, and it is called r^{k+1}.

Every time a device has been relocated after its new predecessor their link becomes permanent and no further moves can break it. Thus, whenever a device is chosen to be relocated after another one, it carries the following chain of fixed edges along with it; this prevents the current choice to destroy the previous ones. To implement this behaviour the procedure should be enriched with a memory, which stores the previous moves.

Figure 2 shows a possible path from r to g consisting of 4 intermediate steps. Table 1 reports at any step the values of P_{g-r}, the chosen move to r^{k+1}, and the fixed edges, for the example in Figure 2. The first move transforms the initial solution into r^1 by relocating 2 after 5; this edge is now fixed (shown in bold in the figure) and this move is stored into the memory, represented by a the dashed box. The second move from r^1 to r^2 relocates the chain $5 - 2$ after 3. Then 7 is relocated after 1 achieving r^3. Finally 1 is relocated after 4 together with its fixed successor 7. The target is finally reached.

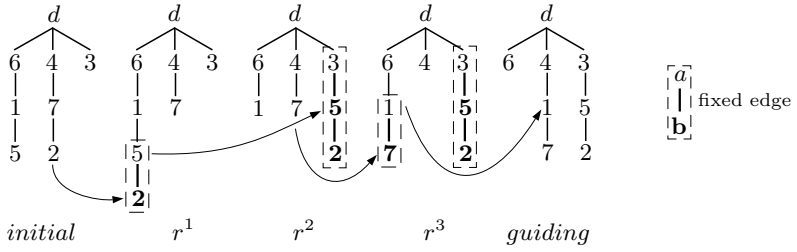

Fig. 2. Feasible References and Target solutions for 7 devices differing on 4 predecessors

Recall that in this real application only solutions with 3 routes are considered to be feasible since only 3 teams are available. So we exclude from P^k the moves that vary the number of routes, i.e., moves that either empty a route or add a new route.

This version of PR moves at most N_{dev} times and calls EPANET at most $\frac{N_{dev}(N_{dev}+1)}{2}$ times. Sometimes the procedure visits the same solution twice or more, in such a case the solution would be evaluated by EPANET more times, wasting precious computing resources. For this reason the solving architecture is

Table 1. Iterations of the PR algorithm for the example in Figure 2

k	$\mathbf{r^i}$	$\mathbf{P^i_{g-r}}$	move	fixed edges
0	r	$\{(1,7),(3,5),(4,1),(5,2)\}$	$(5,2)$	$\{\}$
1	r^1	$\{(1,7),(3,5),(4,1)\}$	$(3,5)$	$\{(5,2)\}$
2	r^2	$\{(1,7),(4,1),\}$	$(1,7)$	$\{(5,2),(3,5)\}$
3	r^3	$\{(4,1)\}$	$(4,1)$	$\{(5,2),(3,5),(1,7)\}$
4	g	$\{\}$		$\{(5,2),(3,5),(1,7),(4,1)\}$

enriched with a cache, which stores the explored solutions and allows for saving a call to EPANET. It is worth noting that the procedure may explore the entire path between *initial* and *guiding* before the maximum number of EPANET calls expires. In such a case, the procedure selects another reference, and iterates over it. The algorithm continues until it reaches the maximum number of EPANET calls or reference solutions.

From now on, we refer to this version as the "routing" PR (PRr).

3.3 The time based variant

Another representation for the mTSP encodes a solution as a vector of activation times [11]. Given the feasible solutions r and g, the indexes of the differing elements is given by $I_{r-g} = \{i \mid r_i \neq g_i\}$. If r equals g then $I_{r-g} = \emptyset$. To transform r into g iteratively, at each step k one element in I_{r-g} should be fixed to its value in g. This decreases by one the distance between r^k and g. Let r^k be the reference vector at the k-th step, $I^k = I_{r^k-g}$, and let $F^k = I_{r-g} \setminus I^k$ be the set of indexes that have been already fixed. The next solution r^{k+1} in the path between r and g is obtained by keeping $r^{k+1}_f = g_f$ for all $f \in F^k$ and fixing the new element $r^{k+1}_i = g_i$ for one $i \in I^k$.

If the remaining elements of r^{k+1} were the same as in r (or r^k) the resulting vector could not correspond to a feasible schedule. So, these elements are allowed to change and are chosen by solving a constrained optimization problem whose constraints depict a mTSP, where the elements in $F^k \cup \{i\}$ are fixed whereas the other (non-fixed) elements are the actual integer variables of the model. The objective is to optimize these variables, so that their values are as close as possible to the ones in r. To do that, the model minimizes the Euclidean distance of the non-fixed elements from r in a manner similar to what it has been been done in [11], i.e., given $i \in I^k$:

$$dist(r^k, r) = \sum_{j \in I^k \setminus \{i\}} |r^{k+1}_j - r_j| \tag{3}$$

Notice that a feasible vector always exists, being g a feasible solution of the program. The neighbourhood of r^k is then defined as follows:

$$\mathcal{N}^k = \{r^{k+1} \mid card(I^k \setminus I^{k+1}) = 1, \text{minimizes (3)}\}.$$

The procedure explores every solution by varying the index $i \in I^k$, and for each i it calls the optimiser to compute a new feasible vector, finally it evaluates the solution by calling EPANET. The solution in N^k having the lowest volume is selected to be the reference solution for the next step. Figure 3 shows, on a graph with 7 devices, how routes change when an additional element becomes fixed; e.g., in r^6 the activation time "(26)", which was (20) in r^5, has been fixed,

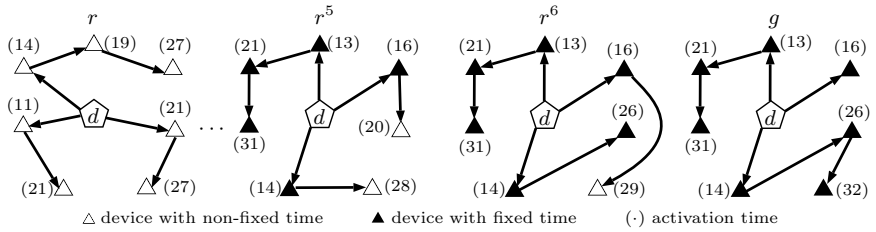

Fig. 3. Feasible routes and activation times for the solution r, r^5, r^6, and g

and the related device is now visited by another route. The minimization of $dist(r^6, r)$ also transforms (28) in r^5 into (29) in r^6, by changing the route of the concerned device; this new activation time is clearly very close to (27) in r.

The constrained optimization program minimizing (3) can be stated by any declarative paradigm, such as Constraint Programming [8], Constraint Logic Programming [21], Answer Set Programming [14, 23], Mixed Integer Linear Programming [25]; so some suitable solvers are: Gecode [13], ECLiPSe [33], DLV [24], Clasp [12], SCIP [1], Gurobi [18].

Since the number of feasible moves decreases at each iteration of at least one unit, the maximum number of EPANET calls is again $\frac{N_{dev}(N_{dev}+1)}{2}$. Also this PR uses a cache to store the explored solution, so it ends up whenever either no more EPANET calls are available, or rs is empty.

Notice how this technique integrates MILP, hydraulic simulators, and PR into the solving architecture; thus, we will refer to it as the "hybrid" PR (PRh).

3.4 Computational results

We extended the experimental campaign introduced in [27] keeping the same experimental platform. In the previous work hybrid path relinking was in general better performing than the route based one in the 20 contamination scenario tested; however, a deeper insight about this result can be obtained by extending the experiments on more scenarios.

The experiments were performed on the Ferrara's hydraulic network, which supplies drinking water to about 130,000 inhabitants. 42 contamination scenarios $(1 \dots 42)$ were tested, 22 more than the previous work [27]. Basing on the techniques proposed in [17], 3 teams of technicians were considered to be available to

operate on 13 hydraulic devices, namely 7 valves and 6 hydrants. The hydraulic simulator we used was EPANET [32], and takes about 5 seconds to evaluate a schedule of the selected devices on each contamination scenario. Even though EPANET is open-source, the simulation procedures and the network specifications are sensitive data for hydraulic engineers and can not be disclosed. It is worth mentioning that, due to the running time required by a simulator call, the results here presented requested about 32 days of computation on AMD Athlon 64 X2 Dual Core Processor 5600+ and 2800MHz.

Genetic algorithms proposed in [11] were allowed a maximum of 500 EPANET calls, and the population was sized to 20 individuals. These values were calibrated in previous works, and the GAs typically converge within the 500 EPANET calls. As mentioned, we observed that the final solution depends on the initial population as typically GA gets stuck at a local optimum, so parallel small sized independent GAs explore the search space better than one big sized GA and can be parallelized. In this study, Path Relinking is tested to explore the region enclosing such solutions.

The hypotheses tested hereby are whether:

1. intensificating over the final population of 10 GAs may improve the local optima;
2. an intensification step may be more effective than an additional independent GA run;
3. given that, hybrid time based GAs outperformed route based ones in [11], likewise we expect (and preliminary proved in [27]) that time based path relinking will dominate the route based one.

To tests such hypotheses the path relinking procedures were started from the final populations of 10 independent GAs. The two PRs were run independently compared to an additional independent GA run, this to be considered a strengthening run of the same first 10. In this way all the approaches are directly comparable in terms of computational resources. The resulting experimental platform is depicted in Figure 4. It shows that the 10 GAs' final populations are merged and filtered to yield the reference set rs; PRr and PRh are started from that rs and return the contaminated water volumes V^r and V^h, respectively; V^g is the best over the solution returned by the additional GA and the former 10 (so the best in rs): basically it is like running 11 independent GAs.

To weaken the randomness, the tests were repeated 10 times on each contamination scenario. To disambiguate, the 10 independent GA runs are intended to be local, with respect to the 10 general runs that are called *global*. PRr, PRh, and the additional GA were allowed 500 EPANET calls each; the total number of simulations is then 6500 for any configuration. Consequently, the total number of EPANET calls is $6500 \cdot 42 \cdot 10 = 2{,}730{,}000$, corresponding to about 32 days of (a single) CPU time. The constrained optimization model minimizing (3) was implemented as a Mixed Integer Linear Programming and solved by Gurobi [18]; its solving time is negligible within the single run (see [11] for some details).

We introduce the following benchmark metrics for the first optimization stage:

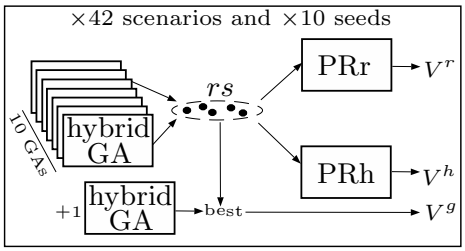

Fig. 4. Graphical view of the experimental platform.

s_c^* $= \min\{rs_c\}$: the best of the c-th trial of 10 GAs;
s^* $= \min_c\{s_c^*\}$: global optimum within the 10×10 GA runs.

Comparing the path relinking outcome with the former assesses whether and how much the intensification step has been effective within the single trial; comparing with the latter instead assesses how effective is the intensification step in finding very good solution with respect to the GAs' ones. To make this comparison we introduce the following metrics for each method of the second optimization stage:

V_c : the contaminated water in output from one of the three method in the c-th trial;
Δ_c $= s_c^* - V_c$: how much contaminated water is saved by the second step with respect to the reference set: as $V_c \leq s_c^*$, $\Delta_c \geq 0$ always holds;
$\overline{\Delta}$: the averaged amount over the trials.

These quantities above are measured in litres (l). Other useful metrics are:

IMP_c $= \{1 \Leftrightarrow \Delta_c > 0, 0 \text{ otherwise}\}$: whether the method improves the local optimum;
IMP $= \sum_c \text{IMP}_c$: how frequently the method can improve the starting solution.

To better explain these metrics Table **??** reports the values for the scenario 7.

Table 2 reports the best volumes $min\{V_c\}$, IMP, and $\overline{\Delta}$ computed by the different approaches for each scenario. To test the first hypothesis, recall we compare how many times the three approaches improve s_c^*, and how many litres they save, that are outcomes of the first optimization step. The additional GA (+1 GA) improved s_c^* only 8 times, but on 8 different scenarios. PRr improved s_c^* overall 54 times and in 21 scenarios never improved it. PRh improved s_c^* overall 272 times, at least twice in every scenarios. So, on average PRr improved s_c^* 1.2 times of 10, while PRh did it 6.5 times; also, that one time PRr improved the local optimum it saved, on average, 20.9 litres of contaminated water, while PRh saved 63.8 litres. This suggests that intensificating pays back, and does it more than an additional (non-intensificating) run of GA, which proves also the second hypothesis.

Table 2. The Table reports for each scenario and in this order: the averaged value $\overline{s}$ and the best value of s^* in the 10 independent *global* runs of 10 GA (so 100 GA runs); it also reports for 10 independent *global* runs of +1 GA, PRh and PRr: the best volume $\min\{V_c\}$, the number of improvements IMP, the averaged improvement $\overline{\Delta}$; last row reports the average of some of these columns. The best values within columns $\min\{V_c\}$, IMP, and $\overline{\Delta}$ are highlighted in bold.

scen	**10 GA** ↓ (sorting key)		**+1 GA**			**PRh**			**PRr**		
	$\overline{s}$	s^*	$\min\{V_c\}$	IMP	$\overline{\Delta}$	$\min\{V_c\}$	IMP	$\overline{\Delta}$	$\min\{V_c\}$	IMP	$\overline{\Delta}$
1	6,022	6,000	6,000	0	0	6,000	7	9.0	**5,997**	**8**	**9.2**
2	6,591	6,575	6575	0	0	**6,574**	**4**	**2.8**	**6,574**	1	0.1
3	7,217	7,170	7,170	0	0	**7,156**	**5**	**14.0**	7,170	1	2.1
4	7,783	7,728	7728	0	0	**7,722**	**9**	**19.5**	7,728	0	0.0
5	10,868	10,672	10,672	0	0	10,672	**7**	41.3	**10,569**	2	**49.9**
6	11,072	10,995	10,995	0	0	**10,979**	**9**	**33.4**	10,995	1	0.4
7	11,229	11,021	11,021	0	0	**10,623**	**7**	**48.9**	10,679	1	4.0
8	11,541	11,336	11,336	0	0	11,336	**7**	**76.5**	11,336	0	0.0
9	11,807	11,638	11,638	0	0	**11,599**	**7**	**45.3**	11,638	0	0.0
10	11,883	11,659	11,659	0	0	**11,592**	**7**	**69.8**	11,659	0	0.0
11	12,301	12,072	12,072	0	0	**12,059**	**6**	**14.3**	12,072	0	0.0
12	12,460	12,260	12,260	0	0	12,260	**5**	**35.1**	12,260	1	0.2
13	12,475	12,289	12,289	0	0	12,289	**6**	**20.6**	12,289	0	0.0
14	12,732	12,698	12,698	1	4.5	12,698	**3**	**15.1**	12,698	1	3.9
15	12,901	12,362	12,362	0	0	12,362	**4**	**5.8**	12,362	0	0.0
16	13,938	13,793	13,793	1	1.5	13,723	**7**	43.5	**13,624**	4	**69.2**
17	14,098	13,748	13,748	0	0	**13,642**	**8**	**72.6**	13,748	0	0.0
18	15,115	14,837	14,837	0	0	14,837	**6**	**46.2**	14,837	0	0.0
19	15,841	15,758	15,758	0	0	**15,692**	**8**	**57.2**	15,758	4	28.9
20	16,991	16,571	16,571	1	3	16,351	**9**	136.8	**15,708**	7	**206.8**
21	19,236	19,162	19,162	0	0	**19,160**	**4**	**8.5**	19,162	0	0.0
22	20,653	19,987	19,987	0	0	19,987	**6**	**89.9**	19,987	0	0.0
23	20,792	20,122	20,122	0	0	20,122	**5**	22.2	20,122	2	**49.9**
24	22,273	22,164	22,164	0	0	**22,105**	**9**	**84.8**	22,164	2	8.3
25	22,792	22,667	22,667	0	0	22,667	**4**	**36.6**	22,667	0	0.0
26	25,138	25,043	25,043	0	0	25,043	**7**	**68.3**	25,043	2	21.2
27	25,280	25,159	25,159	0	0	**25,083**	**9**	**77.1**	25,159	0	0.0
28	26,003	25,410	26,410	0	0	**25,334**	**7**	**71.0**	25,410	1	61.4
29	27,072	26,634	26,634	0	0	26,634	**9**	**173.9**	26,634	0	0.0
30	34,618	34,312	34,312	0	0	34,312	**6**	**134.8**	34,312	0	0.0
31	35,067	34,662	34,662	0	0	**34,536**	**7**	119.8	34,662	4	**135.6**
32	37,050	36,706	36,706	0	0	36,706	**5**	**102.8**	36,706	1	2.2
33	40,121	39,230	39,230	1	20.7	**39,128**	**10**	**214.6**	39,230	4	121.2
34	40,807	40,044	40,044	0	0	**39,727**	**4**	**55.2**	40,044	0	0.0
35	42,019	41,595	41,595	0	0	41,595	**6**	**78.6**	41,595	0	0.0
36	42,443	42,083	42,083	0	0	**42,023**	**7**	**57.1**	42,083	0	0.0
37	44,470	44,286	44,286	1	10.3	**44,188**	**2**	**13.2**	44,286	0	0.0
38	46,452	46,175	46,175	1	2.4	**46,144**	**8**	**137.3**	46,175	0	0.0
39	48,577	48,409	48,409	0	0	48,409	**7**	**45.2**	48,409	0	0.0
40	52,531	52,210	52,210	1	14.7	**52,205**	**5**	**77.1**	52,210	3	57.0
41	77,397	77,232	77,232	0	0	**76,999**	**6**	**122.8**	77,232	2	21.1
42	144,622	144,409	144,409	1	8.4	**144,350**	**8**	**81.7**	144,409	2	24.1
				0.2	1.5		6.5	63.8		1.2	20.9

Notice that one could consider to run either the new 10 (additional) GA or the PRr or the PRh in the same parallel architecture as the first 10 GAs were, considering them as an entire second stage optimization process. In such a case one would select at the end the best solution $\min\{V_c\}$ in order to compare it with s^*. It turns out that the second stage of GAs never improved s^* (which were pretty unlucky), PRr improved s^* in 7 scenarios, and PRh improved it in 25 scenarios; in 13 scenarios out of 42 neither PRr nor PRh did not improve. Consequently, in this context intensification by path relinking is better performing than rerun the first stage again.

As far as the third hypothesis to be tested, we compare how many time PRr improved s^* whereas PRh did not, which happened in 5 scenarios (1, 2, 5, 16, and 20); on the contrary, in 22 scenarios PRh improved s^* whereas PRr did not. We can also compare for how many scenarios PRr got better values of IMP than PRh and vice versa, obtaining that only in scenario 1 PRr improved the local optimum more times than PRh (8 vs 7). Moreover, PRh on average improved s_c^* 6 as many times as PRr (see last row). In terms of contaminated water, PRh saved 3 times the volume than PRr ($63.8\,l$ vs $20.9\,l$). This suggests that PRh more likely finds better schedules than PRr, and decreases the contaminated water more than PRr. Despite of the fact that PRh is generally better performing than PRr, there is not a total dominance, which suggests further investigations.

Figures 6–8 show three boxplots for each contamination scenario. The legend of these charts is provided in Figure 5: the x-axis reports the amount of saved contaminated water Δ_c, while along the y-axis three horizontal boxplots are listed for any scenario: the GA one, then PRr, and finally PRh, from the top to the bottom. Outliers may appear as points.

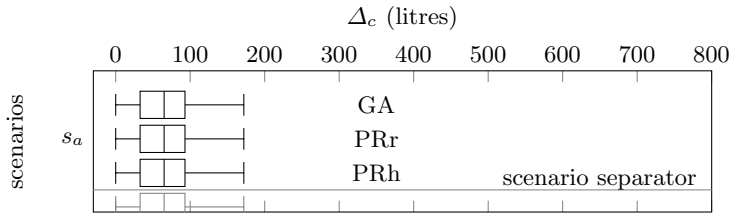

Fig. 5. Legend for charts in Figures 6–8.

These charts show many outliers in many scenarios. This means that many (better) schedules can be found out by path relinking can find particularly good schedules with a low probability; in other words, these schedules are within the spectrum of the solutions explored by the path relinking, but this seldom reaches them. In this simulation-optimization context the simulator is very computational demanding, the issue is then to explore broader region of the search space, which means using simulation calls in a better way: possible perspective in this sense are described in the following section.

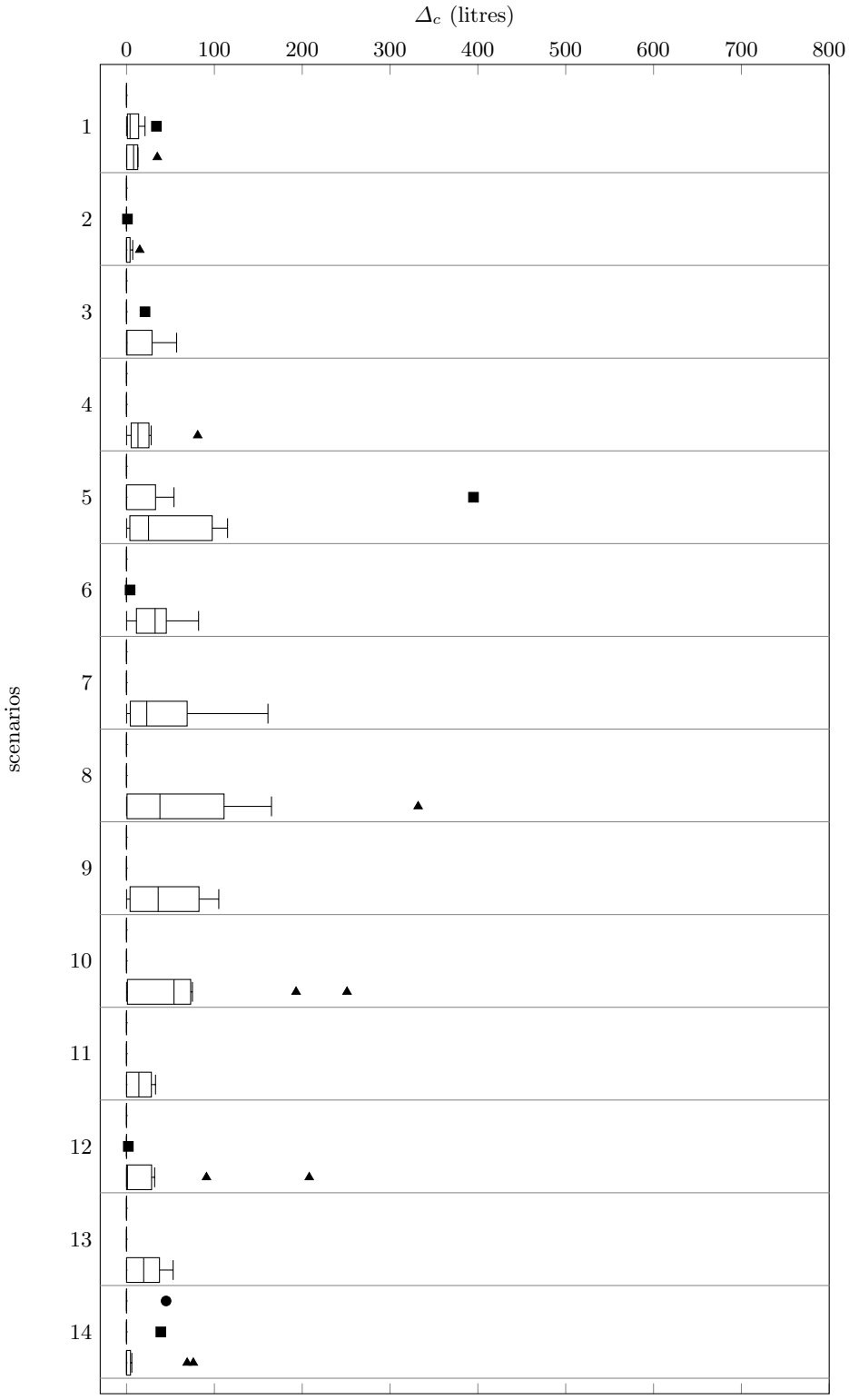

Fig. 6. Boxplots for scenarios from 1 to 14

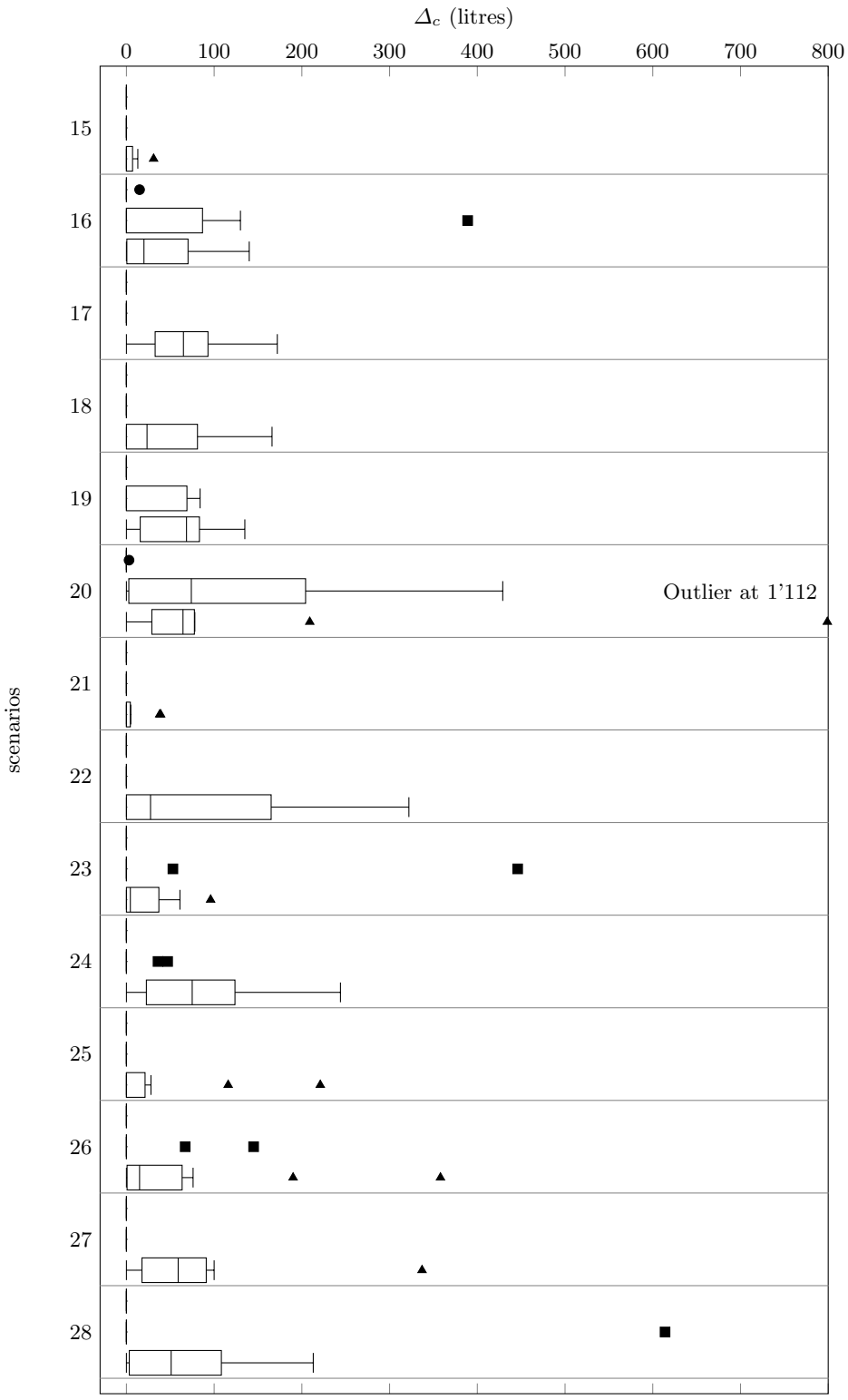

Fig. 7. Boxplots for scenarios from 15 to 28

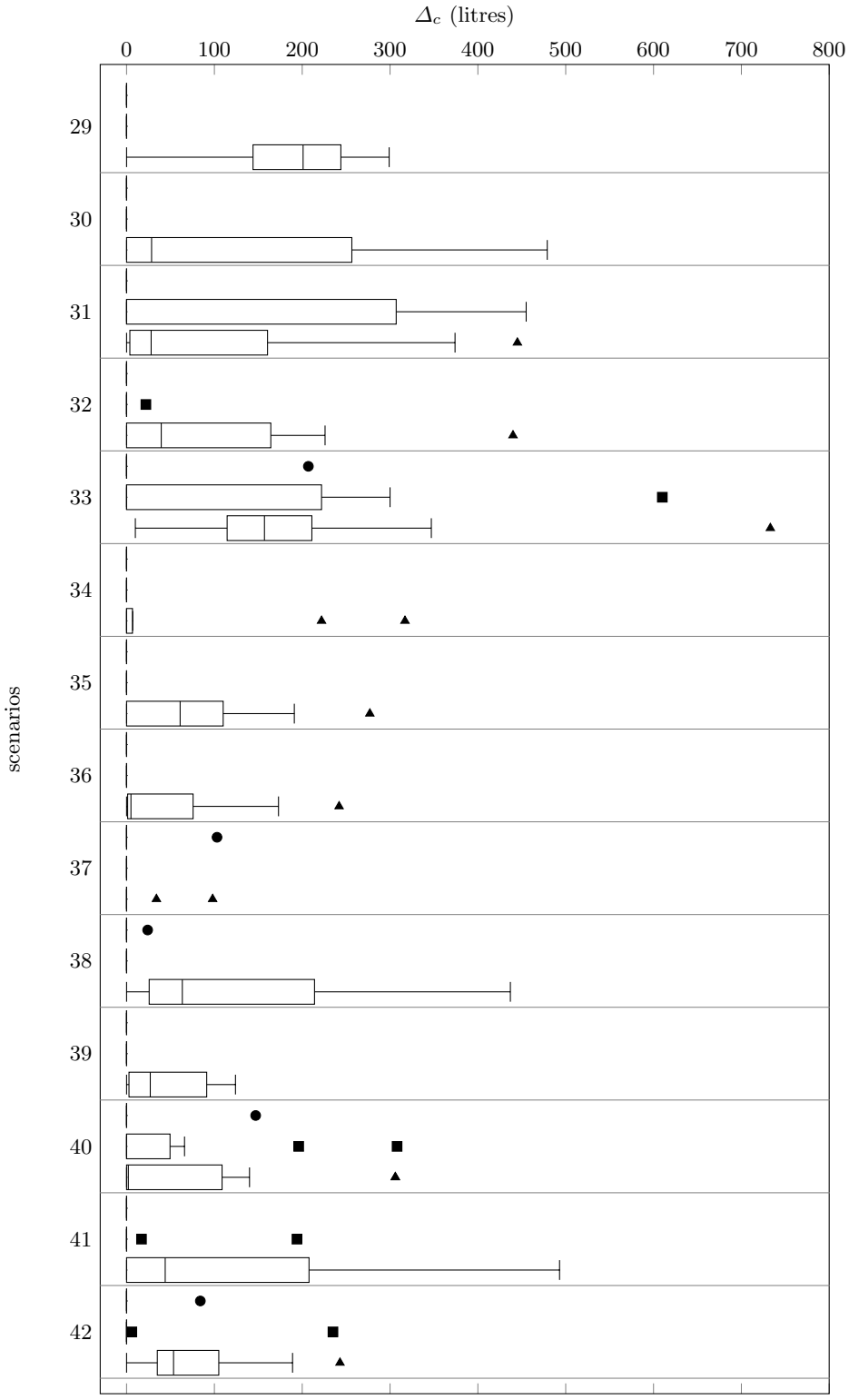

Fig. 8. Boxplots for scenarios from 29 to 42

4 Conclusions and Future Works

This work extends the experimental analysis of an additional intensification step to be performed by path relinking on the local optima in output by 10 parallel genetic algorithms. We tested 42 contamination scenarios, 22 more than the previous work [27]; every test was repeated 10 times to handle with randomness, and the total computing time was about 32 days in a single CPU due to the great deal of effort in hydraulic simulations that was needed to compute the schedules' quality. Results show that intensificating pays back, in fact path relinking saves more contaminated water and more times that an additional independent (thus non-intensificating) genetic algorithm. The time based path relinking, which integrates a m-TSP solver, outperforms route based path relinking in many scenarios: it improves the local optimum more probably and saves much more contaminated water indeed; however, there is no total dominance.

A deeper insight into the data-result shows that path relinking reaches a few times some much better solutions (upper outliers in boxplots); this suggests that its search capability can be further improved to broader explore the search space: in other words, the use of the hydraulic simulator should be subordinate to a smarter strategy and exploit the available knowledge in the current path, in the previous ones, and even in the whole path set simultaneously. Further well-known features of path relinking can be applied to empower its search capability; one idea would be to interrupt (truncate) the search when the previous moves were not good. This can be obtained by implementing the so called *truncated* path relinking. Also, the path direction could be changed making the path relinking uses the best known solution as *initial* instead of *guiding*, in fact better solutions may be closer to the local optimum. Moreover, the starting solution can be updated every time a new local optimum is reached. Also, a path may be explored either with less or more priority depending on the current knowledge and a high level strategy may choose to explore a new portion of a path in the reference set dynamically updating such priority. All these hints are currently under study.

Acknowledgments. We thank Stefano Alvisi and Marco Franchini for assistance with the hydraulic simulator and the instance they developed, and for the fruitful discussions about the hydraulic engineering.

References

1. T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. `http://mpc.zib.de/index.php/MPC/article/view/4`.
2. S. Alvisi, M. Franchini, M. Gavanelli, and M. Nonato. Near-optimal scheduling of device activation in water distribution systems to reduce the impact of a contamination event. *Journal of Hydroinformatics*, 14(2):345–365, 2012.
3. J. April, F. Glover, J. P. Kelly, and M. Laguna. Simulation-based optimization: Practical introduction to simulation optimization. In *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation*, WSC '03, pages 71–78. Winter Simulation Conference, 2003.
4. T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.
5. R. Bent, C. Coffrin, D. Judi, T. McPherson, and P. van Hentenryck. Water distribution expansion planning with decomposition. In *WDSA 2012: 14th Water Distribution Systems Analysis Conference, 24-27 September 2012 in Adelaide, South Australia*, page 305. Engineers Australia, 2012.
6. A. E. Carter and C. T. Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, 175(1):246 – 257, 2006.
7. M. Cattafi, M. Gavanelli, M. Nonato, S. Alvisi, and M. Franchini. Optimal placement of valves in a water distribution network with CLP(FD). *Theory and Practice of Logic Programming*, 11(4-5):731–747, 2011.
8. T. Frühwirth and S. Abdennadher. *Essentials of Constraint Programming*. Springer, 2003.
9. M. Gavanelli, M. Nonato, and A. Peano. An ASP approach for the valves positioning optimization in a water distribution system. *Journal of Logic and Computation*, 2013. In press. doi: `10.1093/logcom/ext065`.
10. M. Gavanelli, M. Nonato, A. Peano, S. Alvisi, and M. Franchini. Genetic algorithms for scheduling devices operation in a water distribution system in response to contamination events. In J.-K. Hao and M. Middendorf, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 124–135. Springer Berlin / Heidelberg, 2012.
11. M. Gavanelli, M. Nonato, A. Peano, S. Alvisi, and M. Franchini. Scheduling countermeasures to contamination events by genetic algorithms. *AI Communications*, 28(2):259–282, 2015.
12. M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):107–124, Apr. 2011.
13. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.
14. M. Gelfond. Answer sets. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 7, pages 285–316. Elsevier Science, 2008.
15. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.
16. J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77 – 95, 2005.

17. M. Guidorzi, M. Franchini, and S. Alvisi. A multi-objective approach for detecting and responding to accidental and intentional contamination events in water distribution systems. *Urban Water*, 6(2):115–135, 2009.
18. Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2014. http://www.gurobi.com.
19. Z. U. Haq and A. A. Anwar. Irrigation scheduling with genetic algorithms. *Journal of Irrigation and Drainage Engineering*, 136(10):704–714, 2010.
20. S. Ho and M. Gendreau. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72, 2006.
21. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programmig*, 19/20:503–581, 1994.
22. A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.
23. N. Leone. Logic programming and nonmonotonic reasoning: From theory to systems and applications. In C. Baral, G. Brewka, and J. Schlipf, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 4483 of *Lecture Notes in Computer Science*, pages 1–1. Springer Berlin Heidelberg, 2007.
24. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499–562, July 2006.
25. R. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Springer US, 1999.
26. R. Murray, W. Hart, C. Phillips, J. Berry, E. Boman, R. Carr, L. A. Riesen, J.-P. Watson, T. Haxton, J. Herrmann, R. Janke, G. Gray, T. Taxon, J. Uber, and K. Morley. US environmental protection agency uses operations research to reduce contamination risks in drinking water. *Interfaces*, 39(1):57–68, 2009.
27. M. Nonato and A. Peano. Path relinking for a constrained simulation-optimization team scheduling problem arising in hydroinformatics. In M. Gavanelli, E. Lamma, and F. Riguzzi, editors, *AI*IA 2015 Advances in Artificial Intelligence*, volume 9336 of *Lecture Notes in Computer Science*, pages 31–44. Springer International Publishing, 2015.
28. A. Peano. *Solving Real-Life Hydroinformatics Problems with Operations Research and Artificial Intelligence*. PhD thesis, University of Ferrara, 2015.
29. C. Prins, C. Prodhon, and R. Wolfler Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4(3):221–238, 2006.
30. A. Rahimi-Vahed, T. Crainic, M. Gendreau, and W. Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of Heuristics*, 19(3):497–524, 2013.
31. M. Reghioui, C. Prins, and N. Labadi. Grasp with path relinking for the capacitated arc routing problem with time windows. In M. Giacobini, editor, *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 722–731. Springer Berlin Heidelberg, 2007.
32. L. A. Rossman. *EPANET 2 users manual*. National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, USA., 2000.
33. J. Schimpf and K. Shen. Ecl$^{\text{i}}$ps$^{\text{e}}$ - from LP to CLP. *TPLP*, 12(1-2):127–156, 2012.
34. H. Simonis. Constraint applications in networks. *Handbook of constraint programming*, 2:875–903, 2006.

35. K. Sörensen and P. Schittekat. Statistical analysis of distance-based path relinking for the capacitated vehicle routing problem. *Computers & Operations Research*, 40(12):3197 – 3205, 2013.