# Path Relinking for a Constrained Simulation-Optimization Team 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.** We apply Path Relinking to a real life constrained optimization problem concerning the scheduling of technicians due to activate on site devices located on a water distribution network in case of a contamination event, in order to reduce the amount of consumed contaminated water. Teams travel on the road network when moving from one device to the next, as in the Multiple Traveling Salesperson Problem. The objective, however, is not minimizing travel time but the minimization of consumed contaminated water. This is computed through a computationally demanding simulation given the devices activation times. We propose alternative Path Relinking search strategies exploiting time-based and precedence-based neighborhoods, and evaluate the improvement gained by coupling Path Relinking with state of the art, previously developed, hybrid Genetic Algorithms. Experimental results on a real network are provided to support the efficacy of the methodology.

**Keywords:** scheduling; neighborhood search; 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

[⋆] Original version at this doi: 10.1007/978-3-319-24309-2_3 integer programming based location models [27] and objective function submodularity [23], the optimal scheduling of devices controlling field irrigation [20] 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 a 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 the 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. Previous approaches [2], [10], and [11] already improved the state of the art in hydroinformatics [18], where schedules were computed by hand: Genetic Algorithms (GAs) can compute better schedules automatically [28]. Hereby, we build upon previous contributions, and we show how solution approaches for such complicated real life problems can largely benefit from the integration of different search paradigms.

In the rest of the paper, first we introduce the problem and recall the solution strategy based on hybrid GA 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 [16], 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 prevent full surveillance. Therefore monitoring is the only viable alternative. In practice, a set of water quality sensors is located on the WDS to test water safety in real time, looking for the presence of potential contaminants [27]. 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, 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 $\tau_{ij}$, 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 optimization problem, the *open m-Travelling Salesman Problem* [4] (mTSP), 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.

All these features motivated our choice of a Genetic Algorithm (GA) 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.

Although we could improve by far and large the best solutions available 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 solutions quality varies 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 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 [16].

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.

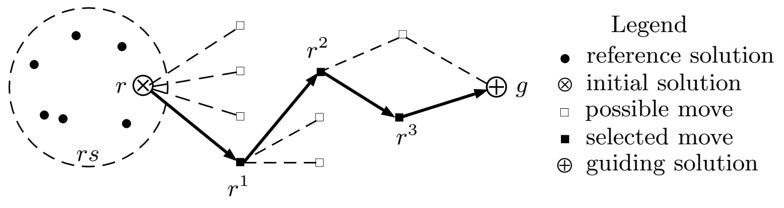

**Fig. 1.** Graphical representation of Path Relinking

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 [17], 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 the distance to the target; thus, different metrics can be combined together to filter properly the initial dataset.

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 [29, 31, 35, 21, 30], 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 wrt the metrics defined for graph representations.

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 $\beta$ and $\gamma$ 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 $\delta$ 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 $\beta$, $\gamma$, and $\delta$ 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

Path Relinking for routing problems works on symbolic representations of routes, in which any route is expressed by an ordered set of visited customers [21, 35, 30]. For 3 vehicles $v_1$,$v_2$, and $v_3$, and 7 customers, namely $c_1, \dots, c_7$, 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 in this mTSP variant. Since the routes have no names, the order of the devices within the routes is the valuable information; moreover, in a route any device precedes only one other, 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 the device $h_p$ precedes $h_s$ in the solution. For example, in the solutions in Figure 2, 3 teams work overall on 7 hydraulic devices, namely $1, \dots, 7$; the *initial* solution $r$ and the guiding $g$ 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 $b$, and vice versa being $card(P_{a-b}) = card(P_{b-a})$. For example, for $r$ and $g$ we have that $P_{r-g} = \{(1,5),(4,7),(6,1),(7,2)\}$, $P_{g-r} = \{(1,7),(3,5),(4,1),(5,2)\}$; so, the distance between $r$ and $g$ is 4.

To get closer to $b$ starting from the configuration of $a$, at least one device $h_s \mid (h_p, h_s) \in P_{b-a}$ 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 the device 2 needs to be relocated right after 5, making $r$ more similar and 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 1 device in $r$ according with $P_{g-r}$.

At the $k$-th iteration, PR has to choose which device $h_s \mid (h_s, h_p) \in P^k_{g-r}$ 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 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 together with its fixed successor 7 after 4. The target is finally reached.

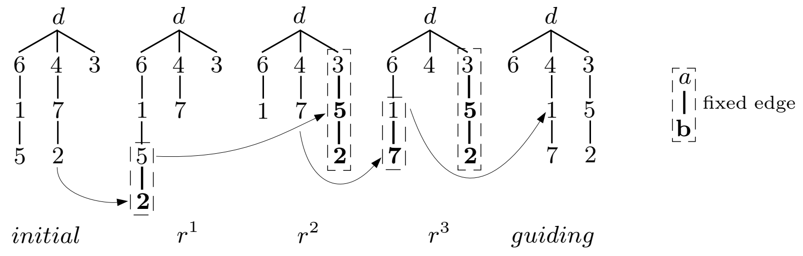

**Fig. 2.** Feasible References and Target solutions for 7 devices differing on 4 predecessors

**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 | $g$ | $\{\}$ | | $\{(5,2),(3,5),(1,7),(4,1)\}$ |

Recall that in this real application only solutions with 3 routes are considered to be feasible. 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 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 was expired. 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, 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 $r_f^{k+1} = g_f$ for all $f \in F^k$ and fixing the new element $r_i^{k+1} = g_i$ 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 chosen by solving a constrained optimisation problem whose constraints depict a mTSP, the elements in $F^k \cup \{i\}$ are fixed, whereas the other (non-fixed) elements are the actual integer variables of the program; the objective is to optimise these variables, so that their values are as close as possible to the ones in $r$. To do that, the program minimizes the Euclidean distance of the non-fixed elements from $r$ [11], i.e., given $i \in I^k$:

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

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, 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 optimisation program minimizing (3) can be stated by any declarative paradigm, such as Constraint Programming [8], Constraint Logic Programming [22], Answer Set Programming [15, 24], Mixed Integer Linear Programming [26]; so some suitable solvers are: Gecode [14], ECLiPSe [33], DLV [25], Clasp [13, 12], SCIP [1], Gurobi [19].

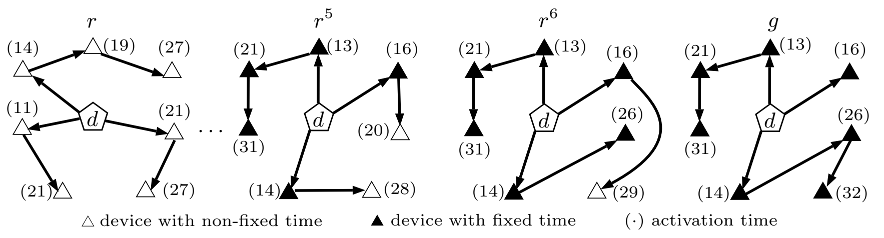

**Fig. 3.** Feasible routes and activation times for the solution $r$, $r^5$, $r^6$, and $g$

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

The experiments were performed on the Ferrara's hydraulic network, which supplies drinking water to about 120,000 inhabitants. 20 contamination scenarios ($A \dots T$) were tested. Basing on the techniques proposed in [18], 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.

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 the GA get stuck on different local optima, so parallel small sized independent GAs explore the search space better than one big sized GA. In this study, Path Relinking is tested to explore the region enclosing such solutions.

The hypothesis tested hereby is that either PRr or PRh may improve the best solution starting from the final populations of 10 independent GAs; in other words, the reference set was built up from the 10 final populations. The two PRs were compared to an additional independent GA run, to be considered a strengthening run of the same first 10. In this way all the approaches are directly comparable. PRr, PRh, and the additional GA were equipped with 500 EPANET calls each; the total amount of calls is then 5500 for any configuration. To weaken the randomness, the tests were repeated 10 times on each contamination scenario. To disambiguate, these runs are considered to be *global*, wrt the 10 *local* GA runs. So we denote the best of the $c$-th trial of 10 GAs as $s_c^*$, while the global optimum is defined as $s^* = \min_c\{s_c^*\}$. The constrained model minimizing (3) was implemented in Mixed Integer Linear Programming and solved by Gurobi; its solving time was negligible.

Table 2 reports the best volumes computed by the different approaches for each scenario. The additional GA (+1 GA) never improves $s^*$; this is quite

**Table 2.** The Table reports for each scenario and in this order: the averaged volume of contaminated water in litres ($l$), the ratio between variance and averaged volume, and the best volume ($l$) for 10 independent *global* runs of 10 GA; it also reports for 10 independent *global* runs of +1 GA, PRr and PRh: the best volume ($l$), the number of improvements, the averaged improvement in $l$; last row reports the average of some of these columns; $min(best)$, $max(impr.\sharp)$, and $max(impr.ave)$ are highlighted in bold.

| scen | **10 GA** | | | **+1 GA** | | | **PRr** | | | **PRh** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ↓ (sorting key) | | | | impr. | | | impr. | | | impr. | |
| | ave $l$ | $\frac{var}{ave}$ | best ($s^*$) $l$ | best $l$ | $\sharp$ | ave $l$ | best $l$ | $\sharp$ | ave $l$ | best $l$ | $\sharp$ | ave $l$ |
| A | 6,022 | 0.04 | 6,000 | 6,000 | 0 | 0 | **5,997** | **8** | **9** | 6,000 | 7 | **9** |
| B | 7,170 | 0.10 | 7,170 | 7,170 | 0 | 0 | 7,170 | 1 | 2 | **7,156** | **5** | **14** |
| C | 10,868 | 1.51 | 10,672 | 10,672 | 0 | 0 | **10,569** | 3 | **49** | 10,623 | **7** | 47 |
| D | 11,229 | 1.16 | 11,021 | 11,021 | 0 | 0 | 11,021 | 0 | 0 | **10,993** | **7** | **44** |
| E | 12,732 | 0.21 | **12,698** | **12,698** | 1 | 5 | **12,698** | 1 | 4 | **12,698** | **3** | **15** |
| F | 13,938 | 0.76 | 13,793 | 13,793 | 1 | 2 | **13,624** | 4 | **69** | 13,723 | **7** | 44 |
| G | 15,841 | 0.22 | 15,758 | 15,758 | 0 | 0 | 15,758 | 4 | 29 | **15,692** | **8** | **57** |
| H | 16,991 | 2.44 | 16,571 | 16,571 | 1 | 3 | **15,708** | 7 | **207** | 16,351 | **9** | 137 |
| I | 20,792 | 7.21 | **20,122** | **20,122** | 0 | 0 | **20,122** | 2 | **50** | **20,122** | **5** | 22 |
| J | 22,273 | 0.39 | 22,164 | 22,164 | 0 | 0 | 22,164 | 2 | 8 | **22,105** | **9** | **85** |
| K | 25,138 | 0.56 | **25,043** | **25,043** | 0 | 0 | **25,043** | 2 | 21 | **25,043** | **7** | **68** |
| L | 35,067 | 1.00 | 34,662 | 34,662 | 0 | 0 | 34,662 | 4 | **136** | **34,536** | **7** | 120 |
| M | 36,706 | 0.52 | **36,706** | **36,706** | 0 | 0 | **36,706** | 1 | 2 | **36,706** | **5** | **103** |
| N | 40,121 | 4.74 | 39,230 | 39,230 | 1 | 21 | 39,230 | 4 | 121 | **39,128** | **10** | **215** |
| O | 42,019 | 1.68 | **41,595** | **41,595** | 0 | 0 | **41,595** | 0 | 0 | **41,595** | **6** | **79** |
| P | 44,470 | 0.34 | 44,286 | 44,286 | 1 | 10 | 44,286 | 0 | 0 | **44,188** | **2** | **13** |
| Q | 46,452 | 1.11 | 46,175 | 46,175 | 1 | 2 | 46,175 | 0 | 0 | **46,144** | **8** | **137** |
| R | 52,531 | 1.47 | 52,210 | 52,210 | 1 | 15 | 52,210 | 3 | 57 | **52,205** | **5** | **77** |
| S | 77,397 | 0.16 | 77,232 | 77,232 | 0 | 0 | 77,232 | 2 | 21 | **76,999** | **6** | **123** |
| T | 144,622 | 0.07 | 144,409 | 144,409 | 1 | 8 | 144,409 | 2 | 24 | **144,350** | **8** | **82** |
| ave | | | | | 0 | 3 | | 3 | 38 | | **7** | **76** |

expected because the additional GA follows the same exploration pattern as any other GA. Anyway, for 8 scenarios, for one $c$ of 10 the additional GA improves $s_c^*$. On average PRr improves $s_c^*$ 3 times of 10, while PRh does it 7 times. For 5 scenarios out of 20 neither PRr nor PRh were able to improve $s^*$; in 4 scenarios (A,C,F, and H), PRr outperforms PRh in terms of global best ($s^*$), and only for one of these scenarios PRh was not able to improve $s^*$. On the contrary, in 11 scenarios PRh improves $s^*$ whereas PRr doesn't. Only in scenario A PRr improves $s^*_c$ more times than PRh. Moreover, PRh on average improves the $s^*_c$ twice as many times as PRr (see last row), decreasing the volume of contaminated water than double the PRr (76 vs 38).

Notice that, the higher is the averaged volume the higher is the outperforming rate of PRh wrt +1 GA and PRr. In fact, from the scenario M onwards, PRh outperforms the others in terms of global best ($s^*$), averaged number of improvements and averaged improvement in volume.

The variance of $s^*_c$, whose normalization over the average is given by $\frac{var}{ave}$ in Table 2, is not correlated to number of improvements the PRs may achieve. In fact, PR is able to improve $s^*$ even for scenarios whose variance is low; this happens mostly when distant local optima have similar quality. Also, since PR's exploration capability grows with the distance between $r$ and $g$, PR should be always coupled to strengthen parallel Genetic Algorithms, even when low variance would suggest that no further improvement is possible.

Finally, despite of the fact that PRh is generally better performing than PRr, there is not a real dominance (A,C,F,H), which suggests to integrate both techniques in future works.

## 4 Conclusions

Genetic Algorithms are used to optimise the scheduling of operations in case of contamination events in Water Distribution Systems [11]; the final populations may contain distant solutions both in terms of similarity and quality. A local search paradigm can improve the solutions by exploiting the knowledge about these local optima.

Two Path Relinking (PR) variants have been developed and tested for a real life hydraulic network, namely the Ferrara's one, to optimise the scheduling of 3 technicians teams over a set of 13 among valves and hydrants, with the aim of reacting to contamination events and minimizing the volume of contaminated water consumed by the users. 20 contamination scenarios were simulated and tested. The hydraulic simulator EPANET was used to compute the volume given a scheduling of the devices; since EPANET takes about 5 seconds to evaluate each solution, we tackled with a computationally intensive consuming simulation-optimisation problem.

A PR was developed to optimise the *routes* of the teams and was named PRr. The other was developed to design directly the activation times of the devices; it was named PRh, from *hybrid*, because it exploits solvers for constrained optimisation programs to compute feasible times; a Mixed Integer Linear Programming implementation was used in this specific case. The two PRs proved to be very effective in improving solutions' quality starting from the final populations of parallel Genetic Algorithms. In the future, more sophisticated PR variants will be tested, e.g., truncated PR, greedy randomized adaptative PR, and others.

***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. M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, Aug. 2012.
14. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.

15. 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.
16. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.
17. 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.
18. 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.
19. Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2014. http://www.gurobi.com.
20. Z. U. Haq and A. A. Anwar. Irrigation scheduling with genetic algorithms. *Journal of Irrigation and Drainage Engineering*, 136(10):704–714, 2010.
21. S. Ho and M. Gendreau. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72, 2006.
22. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programmig*, 19/20:503–581, 1994.
23. 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.
24. 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.
25. 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.
26. R. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Springer US, 1999.
27. 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.
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. 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$^{i}$ps$^{e}$ - 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.