Genetic Algorithms for Scheduling Devices Operation in a Water Distribution System in Response to Contamination Events[*]

Marco Gavanelli, Maddalena Nonato, Andrea Peano, Stefano Alvisi, and Marco Franchini

EnDiF, Università degli Studi di Ferrara
via G. Saragat 1 – 44122, Ferrara, Italy
{marco.gavanelli, maddalena.nonato, andrea.peano, stefano.alvisi, marco.franchini}@unife.it

Abstract. This paper heuristically tackles a challenging scheduling problem arising in the field of hydraulic distribution systems in case of a contamination event, that is, optimizing the scheduling of a set of tasks so that the consumed volume of contaminated water is minimized. Each task consists of manually activating a given device, located on the hydraulic network of the water distribution system. In practice, once contamination has been detected, a given number of response teams move along the network to operate each device on site. The consumed volume of contaminated water depends on the time at which each device is operated, according to complex hydraulic laws, so that the value associated to each schedule must be evaluated by a hydraulic simulation.
We explore the potentials of Genetic Algorithms as a viable tool for tackling this optimization-simulation problem. We compare different encodings and propose ad hoc crossover operators that exploit the combinatorial structure of the feasible region, featuring hybridization with Mixed Integer Linear Programming.
Computational results are provided for a real life hydraulic network of average size, showing the effectiveness of the approach. Indeed, we greatly improve upon common sense inspired solutions which are commonly adopted in practice.

Keywords: Hybrid Genetic Algorithms, Simulation-Optimization, Scheduling

1 Problem Description

The geo-political scenario arising from 9/11 has spurred research concerning infrastructures protection policies and recovery procedures from intentionally induced service disruptions, e.g., because of a terrorist attack. Water distribution systems are among the most vulnerable infrastructures, due to the distributed physical layout of their networks, and to how critical is the commodity they supply: drinking water. People rely on

[*]This paper was published at the conference EvoCOP 2012 [7].

Proceedings of the 19th RCRA workshop on *Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion* (RCRA 2012).
In conjunction with AI*IA 2012, Rome, Italy, June 14–16, 2012.

water quality for a number of crucial activities, such as cooking, washing and bathing in private households, while clear water is essential in operating restaurants, hospitals, and some manufacturing. Adding deadly contaminant into an hydraulic network can rapidly cause huge damage in terms of human losses, since contaminant quickly spreads through the network and is consumed by the users. In this framework, monitoring and promptly recovering is more viable than securing the whole water distribution system, which often has a vast planimetric extent, e.g., a small city network may reach 200*km* and a thousand of pipes and nodes. The common policy followed in the deployment of contamination warning systems consists of installing several sensors along the network, strategically located according to optimization procedures [11], and periodically checking water quality. As soon as a sensor has detected a contaminant, an ad hoc recovery procedure is started, in order to mitigate the impact on the population. Besides population alerting, two kinds of operations can be performed on network devices: opening hydrants in order to expel contaminated water, and isolating pipes by closing their *isolation valves* in order to prevent contaminated water to flow toward densely populated areas. The objective is to minimize the impact on the population, usually measured as the volume of contaminated water consumed by the users during a given period after contamination. This value, which heavily depends on devices activation times, can neither be computed nor approximated by simple analytical calculus, whereas it can be simulated. A simulator such as EPANET [12] takes as input the network configuration, the open/closed status of devices, and the time at which each device is operated, and returns the volume of consumed contaminated water. For real world networks, each simulation may take various seconds of computing time on a modern computer, ($5''$ for a network of about 800 nodes and 1100 main pipes) so that the total number of simulations cannot exceed some threshold to be practically usable, even in an off line procedure such as ours. In most networks, devices can only be operated manually, so teams of workers are dispatched on the network to open hydrants and close valves on site. This introduces significant delays and forbids to operate a large number of devices. The hydraulic engineering literature provides several approaches to select the most suitable subset of devices, given the location of the first alerted sensor: both [1] and [9] propose a multi-objective approach minimizing the number n of operated devices as well as the impact on the population. However, the next major decision concerning the actual schedule of devices activations has never been fully addressed. Indeed, [1] supposes to activate all the selected devices instantaneously and simultaneously, while [9] builds a schedule heuristically according to common sense criteria. However, there is no assurance that this approach gives a (near) optimum scheduling, i.e., a scheduling that minimizes the volume of consumed contaminated water.

This problem has some similarities with the multiple Traveling Salesman Problem ($mTSP$), where m salesmen visit the nodes of a graph minimizing total traveled distance. However, while the $mTSP$ objective function is easily computed, being the sum of traveled distances, ours requires an expensive simulation. Moreover, while $mTSP$'s good quality solutions tend to visit the nodes as soon as possible, in our problem, the early closure of a valve may divert contaminant towards high consumption/demand areas, so that a delay in the schedule sometimes improves the objective function value.

In this work, that extends [2], we propose a genetic algorithm that addresses explicitly the problem of assigning devices to teams (for a given number m of teams) and scheduling the teams operations, in order to minimize the volume of contaminated water consumed by the users. Let us call this problem Response to Contamination Problem (RCP). The genetic algorithm is coupled with an hydraulic simulator, that computes the objective function. We implemented three different chromosome representations and corresponding genetic operators. One representation is original and it is built on the mathematical models developed for the $mTSP$ and for vehicle routing problems (VRP) [13], while the other two come from the literature on the $mTSP$, namely the Two Chromosome and the Two Part representations [5]. The latter has been extended to insert pauses in the schedules while the new one encompasses pauses naturally. We experimentally compare all these representations on the real instance of a medium sized city.

2 Genetic Algorithms for the Scheduling of Operations

Defining a Genetic Algorithm (GA) basically amounts to define the structure of chromosomes, the selection operator, the recombination operators (crossover and mutation), besides fitness measures and termination conditions. In RCP, the evaluation of an individual fitness requires a long hydraulic simulation, so the main obstacle to obtaining good solutions is limited computing time. Therefore, our termination condition is a fixed number of invocations to the hydraulic simulator. Since each call is expensive, we store the input/output data of each call in a sort of caching mechanism with respect to a unique coding of the solution, the activation times vector $t^{\mathscr{F}}$. If the objective function has been invoked before with the same arguments, its value is not re-computed but retrieved from the cache. Thus, the number of invocations is not proportional to the number of generations. Other features common to all the GA families further introduced are: a classical *roulette wheel* procedure for parent selection, an *elitist generational replacement scheme*, mutation of clones, and random generation of the initial population.

2.1 A Genetic Algorithm Based on Sequences

As mentioned, RCP shares the feasibility structure of an $mTSP$ defined on a graph where the mobilization point corresponds to the depot d and each client node to one of the n devices to operate. Then we can borrow from the encodings used for the $mTSP$. One of the first TSP encodings representing the sequence of the visited nodes in a vector extends to the case of m teams by adding a second row, the team identifiers.

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline 3&4&1&2&8&5&7&6\\\hline 1&2&1&3&2&3&2&2\\\hline\end{array} \qquad (1)$$

In the chromosome shown in (1), team number 1 visits nodes 3 and 1 (in this order); team 2 visits 4, 8, 7, and 6, while team 3 visits 2 and 5. This representation is named *two chromosome technique*, we call the related GA $2C$, and the size of its solution space is $n!m^n$ [5]. This encoding, as all those based on permutations, is affected by redundancy which slows down GA's convergence. In fact, the first row of the $2C$ encoding describes a total order on the nodes but it gets decoded into a partial order, which is total only within each route. So, any total order complying with this partial order yields the same activation sequence. So far with the cons. Regarding the pros, this encoding supports simple crossover operators, thanks to the representation into a linear data structure. For example, one can use the *one-point ordered crossover* [8]. Given two parents, f and m, for each integer i in the interval $[1,n]$ two offsprings are generated as follows: the first child inherits the first i columns from f and fills the other columns with the remaining elements taken from m in that order. In the example depicted in (2), i is equal to 4 so the first 4 columns of the child are inherited from f, while the remaining devices, namely 7, 3, 8, and 5, are taken from m in such order, together with the team information.

$$f = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 6&4&1&2&8&5&7&3\\ \hline 1&2&1&3&2&1&2&2\\ \hline\end{array} \qquad m = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 2&7&3&8&4&6&1&5\\ \hline 2&1&3&3&1&2&1&2\\ \hline\end{array} \quad \Rightarrow \quad \underbrace{\begin{array}{|c|c|c|c|}\hline 6&4&1&2\\ \hline 1&2&1&3\\ \hline\end{array}}_{f}\underbrace{\begin{array}{|c|c|c|c|}\hline 7&3&8&5\\ \hline 1&3&3&2\\ \hline\end{array}}_{m} \tag{2}$$

The idea behind this operator is the following. The aim of a good crossover operator is having offspring inherit those features that made its ancestors successful. We have no information about what influences the value of our objective function, lacking a simple analytic formulation: we can only make reasonable assumptions. A possible assumption is that the sequence of activations could influence such value. So, if a sequence is successful, keeping parts of this sequence could make the offspring successful as well. Note that, using a single point crossover, the offspring always inherits the first i elements from one of its parents. This is done on purpose, since devices operated as first strongly influence contaminant spreading, and the first i elements of the sequence are likely to determine which devices are operated first, at least for one team. Figure 1 shows the tree representation of the offspring in (2): in the child tree, the rooted subtree in bold, T_d, comes from f, while the routes of m, after the shrink due to the deletion of the already selected nodes, are randomly appended to T_d according to the team naming adopted in m. Symmetrically, the second child is generated by inheriting the first i columns from m while the remaining devices are activated in the order and by the teams as in f. Each solution (each tree) is associated with an equivalence class of individu-

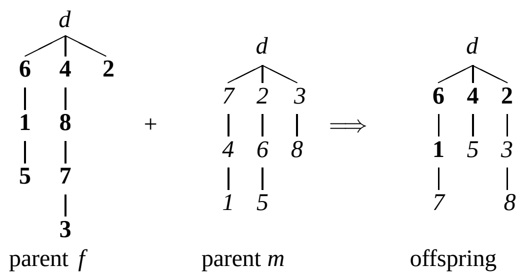

Fig. 1. An example of the tree representation of a crossover

als, each with a different chromosome representation, and this representation impacts on the crossover results. In order to reduce this impact, before crossover we shuffle the columns of each parent while preserving the partial order. In other words, we randomly pick another representative for the same tree in the equivalence class. A further level of redundancy comes from team names; by renaming teams we get different representations of the same solution. To deal with this symmetry, that may generate very different offspring from very similar parents, we adopt a standard team naming approach: the team operating device 1 takes name 1; the team that operates the device with smallest identifier amongst the remaining devices takes name 2, and so on.

2.2 Two-part Chromosome

In [5] a so called two-part chromosome is proposed, with lower redundancy with respect to the other encodings so far proposed in the literature. The permutation part of the chromosome, C_{dev}, made of n integers as usual, is followed by a second part, C_{part}, being a string of m integers summing up to n. Its k^{th} value tells how many elements are part of the k^{th} tour. For example, the same solution depicted in (1) would become (3).

$$\underbrace{\overbrace{\boxed{3}\boxed{1}}^{2}\overbrace{\boxed{4}\boxed{8}\boxed{7}\boxed{6}}^{4}\overbrace{\boxed{2}\boxed{5}}^{2}}_{C_{dev}}\ \underbrace{\boxed{2}\boxed{4}\boxed{2}}_{C_{part}} \tag{3}$$

This way, the size of the representation space is lowered to the order of $n!\binom{n-1}{m-1}$. As in [5], we adopt the above mentioned one-point ordered crossover for the first chromosome part C_{dev}, and a single point asexual crossover (a random rotation) for the second one C_{part}. Both are closed with respect to this encoding and yield feasible solutions.

As already mentioned, in RCP introducing *delays* in the schedule may improve the objective function value. To this purpose, a vector C_{pause} is added to the chromosome assigning a pause to each device, ranging from 0 to an upper bound U. This can be equivalently thought of as the teams moving at *Variable Speed*. For this reason the related GA is named $2P^{VS}$, as opposed to the constant speed version called $2P^{CS}$. In case of variable speed, also the third part C_{pause} is handled by one-point ordered crossover.

While this encoding has a lower redundancy if compared to traditional permutation based encodings, redundancy can not be completely avoided. Indeed, redundancy is inherent into this kind of representation, since the encoding distinguishes among salesmen in the representation space, while they are all identical in the solution space.

In $2C^{CS}$ (i.e., the basic $2C$), $2P^{CS}$ and $2P^{VS}$ GAs, we adopt the same mutation operator, i.e., swapping two columns of the chromosome, applied randomly with given probability. Such probability has a base value of 2%, it is increased of 1% in case of no improvement for 3 consecutive generations, and reset to the base value in case of improvement.

2.3 A Genetic Algorithm Based on Activation Times

The previous encodings support schedule feasibility since they encode $mTSP$ solutions, and any such solution identifies a feasible schedule. However, they do not allow to directly propagate the activation time of a device, that is, the basic piece of information in our problem, which can not be transmitted unless the whole sequence is inherited. A straightforward encoding, which emphasizes the scheduling information, encodes activation times directly in the chromosome, with gene i^{th} modeling activation time of device *i*. Such encoding, being the direct representation of the solution, is redundancy free. The absence of redundancy, however, goes to the detriment of feasibility, which is no longer guaranteed and must be explicitly restored after crossover and mutation. Indeed, a generic vector of activation times does not carry along with it any knowledge of the tours followed in the graph, nor the number of teams, therefore there is no straightforward crossover operator which can preserve feasibility since the encoding itself lacks the necessary information. Consider for example the well known uniform crossover operator (UX), which selects genes from the two parents based on a randomly generated binary mask. A time-based *GA* based on UX may yield vectors spanning the whole space R^n (the most obvious relaxation of the feasible region) but the returned solution may not only be infeasible but also quite different from the closest feasible one. Thus, restoring feasibility after the application of each genetic operator ensures that each individual during the search represents a feasible scheduling, and only feasible schedulings are allowed to reproduce (Algorithm 1). In the following, we introduce a Mixed Integer Linear Programming (MILP) model mapping any vector of activation times to its closest feasible point. It will be used to restore feasibility at every step after the UX crossover, and this approach will be denoted as *UX with a posteriori feasibility restore* (*UXPF*). Furthermore, we extend this idea and integrate the MILP model directly within the genetic operator, giving raise to a second approach denoted as *MILPX*.

Algorithm 1: A genetic algorithm that restores feasibility through a MILP solver

```
$Pop \leftarrow$ generate initial population;
while $not(termination\ condition)$ do
    for $N_{pop}/2$ times do
        Select a pair $(f,m)$ of population individuals;
        $Temp_1 \leftarrow$ Crossover($f$,$m$);
        $Child_1 \leftarrow$call_MILP_solver($Temp_1$);
        $Temp_2 \leftarrow$ Crossover($m$,$f$);
        $Child_2 \leftarrow$call_MILP_solver($Temp_2$);
        randomly apply mutation to individual $I$;
        $I \leftarrow$ call_MILP_solver($I$);
    $Pop \leftarrow \{Child_1, ..., Child_{Npop}\}$ ;
return best;
```

An Integer Programming Model to Restore Feasibility. Let t be a generic vector of activation times. If t is not feasible, i.e., it cannot be obtained by any scheduling of the teams, we propose to repair it by turning it into the feasible point $t^{\mathscr{F}}$ closest to t by norm L_1. As an example, consider a small network with 4 devices plus the mobilization point d, 2 teams and the following traveling time matrix τ:

$$\tau = \begin{array}{c} \\ d \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} d & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} - & 1 & 1 & 1 & 1 \\ 1 & - & 1 & 3 & 1 \\ 1 & 1 & - & 4 & 7 \\ 1 & 3 & 4 & - & 3 \\ 1 & 1 & 7 & 3 & - \end{pmatrix} \end{array}$$

Vectors $m = [1,1,4,8]$ and $f = [2,5,1,1]$ model feasible schedules but the UX operator, by using the binary mask [1,1,0,0], yields the infeasible child $t = [1,1,1,1]$; the restoring procedure returns $t^{\mathscr{F}} = [2,1,1,3]$ as the closest feasible vector, which is indeed at 3 units distance from t by L_1.

Several MILP models can be adopted to find $t^{\mathscr{F}}$, building on those developed for the *mTSP* [4] and routing problems in general, among which the following *2-index flow-based* formulation [13]. The constraints extend the *mTSP* model with traveling times information, the objective function minimizes the distance from t. The unknowns are:

- X a matrix $(n+1)\times(n+1)$ of 0-1 variables. $x_{ij} = 1$ iff j is activated right after i by the same team; i is activated first by its team iff $x_{di} = 1$; $x_{ii} = 0 \forall i$ (no self loop arcs).
- $t^{\mathscr{F}}$ a vector of $n+1$ activation times; $t_i^{\mathscr{F}}$ is the time at which device i is activated, and $t_d^{\mathscr{F}}$ is the departure time from the depot d.
- δ a vector of n differences: it is defined as $\delta_i = t_i - t_i^{\mathscr{F}}$.

The input parameters are:

- t a vector of n ideal activation times.
- τ a matrix $(n+1)\times(n+1)$; τ_{ij} represents the time that a team takes to move at a given constant speed from the location of device i to that of device j.

The constraints:

$$\forall i \in \{1..n\} \qquad t_i^{\mathscr{F}} \geq \tau_{di} \tag{4}$$

$$\forall i \in \{1..n\} \qquad \delta_i = t_i - t_i^{\mathscr{F}} \tag{5}$$

$$t_d^{\mathscr{F}} = 0 \tag{6}$$

$$\sum_{i\in\{1..n\}} x_{di} = m \tag{7}$$

$$\forall i \in \{1..n\} \qquad \sum_{j\in\{1..n\}\,\cup\, d} x_{ij} = \sum_{h\in\{1..n\}\,\cup\, d} x_{hi} \tag{8}$$

$$\forall i \in \{1..n\} \qquad \sum_{j\in\{1..n\}\,\cup\, d} x_{ij} = 1 \tag{9}$$

$$\forall i \in \{1..n\} \qquad t_i^{\mathscr{F}} \leq M + x_{di}(\tau_{di} + U - M) \tag{10}$$

$$\forall i,j \in \{1..n\} \qquad t_i^{\mathscr{F}} + \tau_{ij} \leq t_j^{\mathscr{F}} + (1-x_{ij})M + x_{ji}(\tau_{ij} + \tau_{ji} + U - M) \tag{11}$$

Constraint (4) says that device i can be activated no earlier than the time it takes to reach it from d. Eq. (5) is the definition of δ. Teams leave the depot at time 0 (6). All teams depart from the depot (Eq. (7)). For each node i, the total number of teams arriving to i is equal to the number of teams leaving i, (8), the so called flow balance constraints. All nodes except d are visited exactly once (9). Constraint (10) is the linearization of the implication $x_{di} = 1 \Longrightarrow t_i^{\mathscr{F}} \leq \tau_{di} + U$ (where M is a sufficiently large positive number and U the upper bound for the potential pause before each activation), so that, together with (4), it imposes that the starting time of the first devices be equal to their traveling time from d plus a potential pause. Constraint (11) links the activation times $t^{\mathscr{F}}$ to the ordering between devices given by matrix X; indeed, (11) linearises the implications:

$$x_{ij} = 1 \implies t_i^{\mathscr{F}} + \tau_{ij} \leq t_j^{\mathscr{F}} \qquad x_{ij} = 1 \implies t_i^{\mathscr{F}} + \tau_{ij} + U \geq t_j^{\mathscr{F}}.$$

If $U = 0$ then (11) imposes that the arrival time at device j equals the starting time from i plus the traveling time from i to j, thus implementing the constant speed variant of the time-based *GA*. Conversely, if $U > 0$ the same constraint allows for a maximum pause of U, thus implementing the variable speed variant of the algorithm. The objective function associated to problem (4-11) is the minimization of $||\delta||_1$, i.e. $min\left(\sum_{i\in\{1..n\}}|\delta_i|\right)$. To linearise this function, we introduce new unknowns δ^+ that represent the absolute value of δ, and minimize their sum.

Tighter Integration GA-MILP. Restoring feasibility after crossover may yield children quite different from their parents, since feasibility restoring could disrupt those patterns responsible for parents' fitness. For this reason, we moved the call to the MILP solver *inside* the crossover operator, giving raise to a new operator that we call *MILPX*. In this way, *MILPX* generates directly a new individual proven to be feasible and, at the same time, resembling the most to its parents among all their feasible children.

More precisely, given the chromosomes of the mating individuals $f \equiv (f_1, \dots, f_n)$ and $m \equiv (m_1, \dots, m_n)$, we generate the child c that minimizes the quantity

$$\sum_{i=1}^{n} min(|c_i - f_i|, |c_i - m_i|).$$

Stated otherwise, we can consider each chromosome as a point in a n-dimensional space. The two chromosomes f and m of the mating individuals define a hyper-parallelepiped that has m and f as two verteces, and with sides parallel to the coordinate axes (Figure 2). The MILP solver selects the feasible point in the n-space closest to

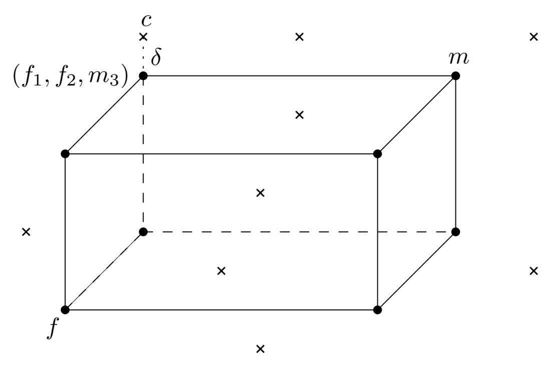

Fig. 2. Graphic representation of the crossover in a 3D space. Crosses represent feasible points, m and f are the mating individuals; c is the closest feasible point (at distance δ) to a vertex of the parallelepiped.

any vertex of the hyper-parallelepiped. In this way, if there exists a feasible point in the n-space that inherits each coordinate from one of the two parents, it will be generated (or, if there exist more points with such feature, one of them is definitely generated as a spawn). Otherwise, the feasible point closest to one of such points is the spawned individual. This is implemented by slightly modifying the MILP model (4-11), by introducing a vector of unknowns W to range on the verteces of the hyper-parallelepiped; $w_i = 1$ iff the i-th coordinate of child c is inherited from f (i.e., $c_i = f_i$) and $w_i = 0$ otherwise (if $c_i = m_i$). The definition (5) of the displacement δ becomes (12)

$$\forall i \in \{1..n\} \qquad \delta_i = f_i w_i + m_i(1 - w_i) - t_i^{\mathscr{F}} \tag{12}$$

Summing up, for the *GA* based on activation times, we propose two crossovers, $UXPF$ and $MILPX$, which can be used within the same algorithm, being invoked with different probability, yielding the so called *time-based Hybrid GAs*. Finally, mutation is applied when a generated offspring already belongs the current population (a clone), and consists of swapping the activation time of two devices, restoring feasibility if necessary.

3 Computational Results

We applied the presented *GA*s to the water distribution network of Ferrara, Italy, population 130,000. A previous work on the same network [9], selected the set of devices to be operated after contamination detection by way of a multi-criteria *GA*, targeting both minimal number of devices and minimal volume of consumed contaminated water, supposing to have as many teams as devices, all departing at the same time. From the Pareto front provided in [9], a point associated with a good trade-off was selected, yielding the $n = 13$ devices to be operated. Commonly, the response procedure starts as soon as a sensor raises the alarm. As stated in [9], an alarm event detects a dangerous toxicity plausibly due to several contamination's locations and times; in our case, 42 contamination scenarios exist which can be simulated and then optimized. Among those, we selected the 5 the most equally spread w.r.t. to the objective function value associated to the scheduling computed according to the *as soon as possible* criterion (ASAP). This scheduling, in turn, is obtained by solving a MILP model for the $mTSP$ with constraints (4), (6-11), and $U = 0$, minimizing the maximum among the devices activation times $\{t_i^{\mathscr{F}}, i \in 1..n\}$, which is also called the *makespan*.
CBC COIN-OR [6] is the MILP Solver used to tackle the optimization problems in the Hybrid *GA*s and to compute the makespan. The hydraulic simulations were performed by EPANET [12], an open-source software developed by the U.S. Environmental Protection Agency (EPA). Each simulation requires on average about 5 seconds. Since we use a cutoff of 500 invocations to the hydraulic simulator, the average computational time of each *GA* is 5×500 seconds. Other parameters are the population size $N_{pop} = 20$, and the team number $m = 3$ (a value set by the managers of the utility company operating the Ferrara network). With these parameters, CBC running time is negligible w.r.t. EPANET.
Overall, we ran 13 *GA*s. The first 10 belong to the time-based Hybrid *GA*s family (section 2.3) and differ from each other regarding speed configuration, i.e. constant speed (CS) and variable speed (VS), and the chance of using the $MILPX$ method rather than $UXPF$ as the crossover operator at the current iteration. More specifically, we tested five $MILPX$ probability values, namely $\{0, 25, 50, 75, 100\}$%. The other three *GA*s belong to the sequence-based family (section 2.1), namely, $2C^{CS}$, $2P^{CS}$, and $2P^{VS}$ *GA*s.

For each scenario, we run each *GA* 10 times. Each *GA* at each run shares the same initial population as the other *GA*s. Fig. 3 shows, for each *GA* in ascending order, the average of the objective function values achieved in the 5×10 runs.

All the time-based Hybrid *GA*s rank first, preceding all the sequence-based ones. The last stack, at the far right of the histogram, represents the average cost of the five solutions returned by the ASAP policy computed on the 5 chosen scenarios.

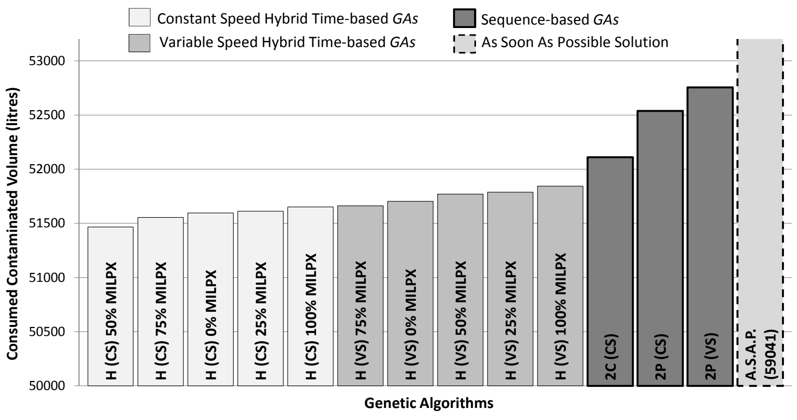

Fig. 3. Average of all the optimal candidates for each *GA* configurations and of the A.S.A.P. known solution

The histogram in Figure 3 allows us for several important considerations:
(a) the ASAP policy neither provides a lower bound for RCP nor a near-optimal solution, since its average volume of consumed contaminated water is much higher than any proposed *GA*s.
(b) Time-based encodings find better solutions than those based on sequences; this is probably due to the fact that the time-based Hybrid *GA*s work in the same solution space of the hydraulic simulator.
(c) Constant speed encodings find better solutions than variable speed encodings; this may be due to the fact that the search space of the variable speed encodings is quite larger due to pauses. Nevertheless, all variable speed Hybrid *GA*s outperform the constant speed sequence-based *GA*s.
(d) For both variable speed and constant speed, a mixed *UXPF* and *MILPX* policy for the Hybrid *GA*s finds, on average, better solutions than totally unbalanced policies.

The histogram in Figure 3 shows that hybrid MILP-*GA*s based on a time representation have on average a better performance with respect to the encodings known in the literature. However, in principle this result could be due to luck: indeed, we can only experiment on a limited number of instances and all algorithms are based on some randomness. For this reason there exists a probability that the new time-based algorithm is worse than the others, despite of its better performance in the finite number of the performed experiments. In order to disprove such conjecture, one should use a *significance* *test* [3]. We apply it to the five algorithms described earlier, namely the H^{CS}, H^{VS}, $2C^{CS}$, $2P^{CS}$ and $2P^{VS}$. For the first two, we selected the best configuration with respect to the percentage of the two crossovers, i.e., 50% for the H^{CS} and 75% for the H^{VS}. A common test used to compare multiple algorithms is the Friedman test [10]. In our case it affirms, with a confidence less than 10^{-4}, that there are some algorithms which perform significantly differently. In order to find the significantly different pairs, one should use the so-called *post-hoc analysis*; we adopted the Nemenyi procedure [10], that consists of pair-wise tests within the whole set of groups. For each pair of algorithms, Table 4 reports the confidence level (the so-called p-value) when assuming the two algorithms have different behaviour. As we can see, such confidence is very low, and in many cases below 10^{-4}.

	H^{CS}	H^{VS}	$2C^{CS}$	$2P^{VS}$	$2P^{CS}$
H^{CS}	1	0.0193	< **0.0001**	< **0.0001**	< **0.0001**
H^{VS}	0.0193	1	0.0499	< **0.0001**	< **0.0001**
$2C^{CS}$	< **0.0001**	0.0499	1	0.0054	**0.0016**
$2P^{VS}$	< **0.0001**	< **0.0001**	0.0054	1	0.7043
$2P^{CS}$	< **0.0001**	< **0.0001**	**0.0016**	0.7043	1

Fig. 4. The $p-value$ for each pair-wise Nemenyi test. The bold $p-values$ are less than α^*.

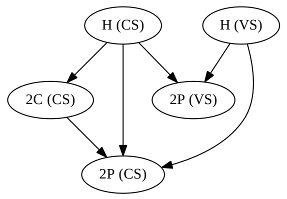

Fig. 5. The dominance graph

However, although single confidence levels are low, the probability of having *at least one error* increases with the number of comparisons (i.e., $N_{comp} = \frac{5\times 4}{2} = 10$). Conventionally, p-values are considered significant when they are below $\alpha = 5\%$. In order to ensure that the whole table contains no errors with p-value below α, Bonferroni [10] suggests to take as significant in Table 4 only those pairs for which the significance is below $\alpha^* = \alpha / N_{comp} = 0.05/10 = 0.005$.

Results in Fig. 4 are graphically depicted in Fig. 5, where an arrow from algorithm A to B means that algorithm A dominates B with a p-value below 0.5%. Accordingly with the histogram in Fig. 3, the dominance graph in Fig. 5 confirms that the Hybrid time-based *GA*, with a very little margin of error, achieves lower volumes of consumed contaminated water with respect to all sequence-based *GA*s.

4 Conclusions

In this study, we addressed an important problem in the security of water distribution systems: the near-optimal planning of the response to an event of contamination.

We tackled the problem by way of genetic algorithms which optimize the value of a black-box objective function, computed through a hydraulic simulator. We implemented two crossover operators taken from the literature on multiple traveling salesman problem, then we proposed and implemented two new crossover operators that exploit a mixed-integer linear programming solver, obtaining a hybrid GA-MILP algorithm.

We ran an extensive experimentation, in which we compared 13 variants of the various algorithms on 5 scenarios for 10 runs each. Considering that each invocation of the black-box function takes about 5 seconds on a modern computer and that we used a cutoff of 500 invocations, we have a total computing time of about 19 days.

All the proposed *GA*s improve on the common sense inspired solution. This confirms that the actual scheduling times impact on the solution value and should be explicitly taken into account by any recovery procedure. Comparing their average behaviour, we observed that the new, hybrid, algorithms outperform all the others. A significance test confirms this result, with a confidence level below 5%.

In future work, we plan to experiment on other scenarios and additional devices, and to test the effect of variable speed in 2*C GA*.

Acknowledgements This work was partially supported by EU project *ePolicy*, FP7-ICT-2011-7, grant agreement 288147. Possible inaccuracies of information are under the responsibility of the project team. The text reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained in this paper.

References

1. L. Alfonso, A. Jonoski, and D. Solomatine. Multiobjective optimization of operational responses for contaminant flushing in water distribution networks. *Journal of Water Resources Planning and Management*, 136(1):48–58, 2010.
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*. In press.
3. T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010.
4. T. Bektas. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.
5. 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.
6. J. Forrest and R. Lougee-Heimer. *CBC User Guide*. Computational Infrastructure for Operations Research, http://www.coin-or.org/Cbc/cbcuserguide.html.
7. 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. 10.1007/978-3-642-29124-1_11.
8. D. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
9. 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.
10. M. Hollander and D. Wolfe. *Nonparametric Statistical Methods, Second Edition.* Wiley, 1999.

11. 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.
12. L. Rossman. *EPANET 2 users manual*. National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, USA.
13. P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.