Scheduling Countermeasures to Contamination Events by Genetic Algorithms

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

ENDIF,
Università di Ferrara,
Via Saragat, 1
44122 Ferrara
Italy
E-mail:
{*marco.gavanelli,maddalena.nonato,andrea.peano,*
stefano.alvisi,marco.franchini}*@unife.it*

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.
Extensive 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

AI Communications
ISSN 0921-7126, IOS Press.

1. Introduction

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. Threats that used to be perceived as low risk, and therefore used to be disregarded, have now become part of the list of potential events that may put at risk national security, not only in the US but all over the western world.

Water distribution systems (WDSs) are among the most vulnerable infrastructures. They are greatly exposed to the risk of contamination by chemical and biological agents due to the physical layout of their networks and to the sensitivity to contamination of the commodity they supply: drinking water. People rely on water quality for a number of crucial activities, such as cooking and washing in private households. Clear water is essential in operating restaurants, hospitals, and farmings. Most industries rely on clear water availability for their functioning, not to mention the use of water in the food supply chain. The potential damage due to water contamination is not only economical. Adding deadly contaminant into an hydraulic network can cause human losses, since contaminant quickly spreads through the network and population alerting strategies may not entirely ward off users' water consumption. For these reasons, WDSs are sensitive targets for terrorist attacks. Accidental contamination events, though, are not of minor concern, and they may happen in the most unpredictable way. Real examples include the case of a fire truck connected to a fire hydrant which injected aqueous firefighting foam into the neighborhoods pipes, or the unintentional dispersion through the network of the salmonella bacteria, originally present in a contaminated storage tank in disrepair.

WDSs usually have a vast planimetric extent, e.g., a small city network may reach $200km$ and a thousand of pipes and nodes. Their wide extension and sparse topology makes WDSs difficult to secure either by inspection or by forbidding access. Indeed, almost every user connection provides an unprotected access to the local public WDS. In this framework, monitoring and promptly recovering is more viable than securing the whole WDS. Much effort has been devoted by the scientific community to the development of sensor based Contamination Warning Systems (CWSs). The common policy followed in the deployment of CWSs consists of installing several sensors along the network, strategically located according to optimization procedures [24]. Sensors periodically check water quality. As soon as a sensor has detected a contaminant, countermeasures are implemented in order to mitigate the impact on the population.

The complete shut down of the network is usually to be avoided due to its many drawbacks. For example, fire-fighting capabilities must always be ensured, and customers with special needs which rely on continuous supply may be harmed. If the network is well districted, it may be possible to isolate the affected part of the network without failing to provide water supply to the entire community. However, when the network is of large size and it is not divided into district metered areas, the water utility is faced with the choice of leaving out of service an entire city or undertake some action on the network devices, which can alter the water flow and modify the contaminant spreading, in order to achieve contaminant isolation, containment, and flushing.

2. Problem Description

Two kinds of operations can be performed on network devices to realize system flushing and system isolation: the former is achieved by opening hydrants in order to expel contaminated water, while the second consists of isolating pipes by closing their *isolation valves*. In most WDSs, devices must be operated manually by teams of technicians who are dispatched on the network to open hydrants and close valves on site. This introduces significant delays on the operations and forbids to operate a large number of devices in case of a limited number of teams. Devices selection is itself a challenge, largely addressed in the hydraulic engineering literature (see section 3). Those studies also show that the time at which a device is operated strongly impacts the pattern of contaminant spreading.

The problem is to determine the feasible scheduling of the devices operations which minimizes the impact on the population. This objective is usually measured as the volume of contaminated water consumed by the users during a given period after contamination.

A first issue concerns the objective function computing time. The volume of consumed contaminated water depends on the time at which each device is activated, but it can neither be easily computed nor approximated by simple analytical calculus, whereas it can be simulated. An hydraulic and quality simulator such as EPANET [25] takes as input the network configuration, the open/closed status of the devices, and the scheduling, i.e., the time at which each device is operated, and returns the volume of consumed contaminated water. EPANET is a discrete event-based simulator, so the length of the simulation period and the size of the time step used in the simulation affect the computational burden of the process. In this case, the simulation period must be long enough to span all the interval going from the time the alarm is raised to the time when contaminant disappears from the network, i.e., its concentration falls below the danger threshold at any demand point along the network. In this study, we are concerned with those substances which can cause human death even when present at low concentration. Therefore, the period to be considered spans several hours after the contamination event. At the same time, though, the size of the simulation time step must be short enough to capture all meaningful variations of water flow and users demand at demand nodes over time. For results to be reliable in case of real world networks, each simulation may take various seconds of computing time on a modern computer, such as $5''$ for a network of about 800 nodes and 1100 main pipes. As a consequence, whatever the solution approach is, the total number of calls to the simulator cannot exceed some threshold to be practically usable, even in an off line procedure such as ours.

A second issue concerns the lack of a priori information about the features of a good schedule.

Unfortunately, scheduling the devices according to common sense inspired criteria is of no use in reducing the volume of consumed contaminated water [19]. In particular, *the sooner the better* principle does not hold once the set up time required for the teams to get ready has elapsed, since the contaminant is already flowing through the network. Therefore, the objective is not to operate all devices within the minimum makespan, as common sense would suggest.

In this problem, there is no rule of thumb to predict the features of a good schedule, because of the complicated hydraulic laws that rule contaminant spreading. Valves, when closed, interrupt water flow and may isolate a pipe. Hydrants, when open, attract water, alter water flow, and expel large water quantities. These actions, when combined, potentially redirect contaminant away from heavy consumption sectors of the network, decrease its concentration below the critical threshold, or prevent contaminated water to flow toward densely populated areas. The outcome of these interactions can not be anticipated simply by looking at the operation times, and there is no way to discriminate between a good and a bad schedule without simulation.

A third issue concerns the feasibility of the schedule to be computed. The devices assigned to the same team of technicians are activated one at a time, sequentially, at different times, since travel times along the route, from a device to the next, are not negligible.

A vector $t^{\mathcal{F}}$, representing the activation times of each device, is feasible if and only if there exists a partition of the n devices into the m teams and a total order on each subset, such that the following conditions hold. Having set to 0 the departure time $t_d^{\mathcal{F}}$ from the mobilization point d, then the operation time $t_j^{\mathcal{F}}$ of device j, if operated by the same team and immediately after device i, with no idle time in between, must verify the equation

$$t_i^{\mathcal{F}} + \tau_{ij} = t_j^{\mathcal{F}} \tag{1}$$

where τ_{ij} is the traveling time required to cover the distance $dist_{ij}$ from the location of device i to j, with respect to a given constant speed. In case of variable speed, $t_j^{\mathcal{F}}$ can not exceed $t_i^{\mathcal{F}} + \tau_{ij}$ of more than a given quantity, so that $dist_{ij}/(t_j^{\mathcal{F}} - t_i^{\mathcal{F}})$ is below the maximum speed. All schedules complying with these requirements are feasible. The search for an optimal schedule must take such constraints into account.

Summing up, this problem poses many challenges from a mathematical programming point of view. The objective function value is obtained by a computationally expensive simulation. The lack of analytical representation does not provide gradient related information. The feasible region has a combinatorial structure, so that, on the one hand, solutions enumeration is no viable even for small instances, on the other hand, the feasible region is not continuous, and a schedule with a good objective function value may be of no use if quite far from any feasible schedule. No relaxation on either sides, i.e., objective function relaxation or feasibility constraints relaxations, provides a meaningful lower bound. Upper bounds obtained by common sense inspired solutions can be quite loose. In this framework, meta-heuristics appear a viable tool for tackling this problem.

[5] discusses in depth the advantages of meta-heuristics for solving optimization-simulation problems in combinatorial optimization. Meta-heuristics are robust with respect to the kind of function to be optimized and are able to exploit the combinatorial structure of the problem to build search trajectories in the feasible region. Concerning the choice of a meta-heuristic for the current problem, recall that simulation is the only way to evaluate the quality of a schedule, and we can not evaluate a priori the effect of small modifications with respect to a given schedule, such as postponing or anticipating the operation time of a single device. Therefore, there is no computational saving in searching within a neighborhood of the current schedule rather than far away from it. Moreover, in local search based methods, adjacent solutions along the search trajectory are generally close to each other, due to the neighborhood mechanism. Therefore, a reduced number of function evaluations would not allow such methods to move sufficiently far from the starting solution. In such cases, global searches such as evolutionary algorithms are often preferred to neighborhood based local searches, for their ability to explore a wider part of the feasible region with a limited number of solution evaluations. Such considerations lead us to selecting Genetic Algorithms (GA) as the solution approach.

In this work we propose three genetic algorithms that address 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). RCP is introduced in section 1, and its challenges discussed in this section. Section 3 discusses related works. The genetic algorithms we present is coupled with an hydraulic simulator, that computes the objective function. We implemented three different chromosome representations and corresponding genetic operators. Common features are described in section 4. The *time-based* representation introduced in section 7 is original and it is built on the mathematical models developed for the multiple Traveling Salesman Problem ($mTSP$), while the other two come from the literature on the genetic algorithms for the $mTSP$, namely the *two chromosome* representation described in section 5 and the *two part chromosome* representation presented in section 6. We experimentally compare all these representations on the real instance of a medium sized city. Results are presented and discussed in section 8. Conclusions are drawn in section 9.

This work extends [4] by providing the experimental comparison with the literature inspired encodings and by tackling the variable speed variant. This variant has been modeled by allowing the team to pause, up to a maximum time, before operating a device. To this purpose, the Two Chromosome and the Two Part representations have been extended to insert pauses in the schedules, while the time-based representation encompasses pauses naturally. A preliminary report of this research was presented in [15]. With respect to that study, in this paper we extend the computational campaign by providing the variable speed variant of the Two Chromosome encoding and by performing an extended computational comparison over a larger set of scenarios and a much higher number of runs of the GAs. We also provide experimental data supporting the calibration of the population size, which was missing in [15]. Pros and cons of using high performance MILP solvers in the framework of the time-based representation are also discussed. Finally, a new section devoted to the literature has been added.

3. Related Works

RCP is a multi-disciplinary problem, as it involves issues related to hydraulic engineering, simulation, and combinatorial optimization. Related works can thus be found in all these disciplines.

The hydraulic engineering literature provides several contributions that report the use of optimization techniques, up to various degrees, to tackle two problems strongly related to RCP, i.e., the sensor location problem and the identification of the devices to be activated once contamination has been detected. These two problems are solved in this order, before solving RCP.

Concerning sensor location when designing a CWS, [24] proposes a MILP model which is close to the p-median facility location problem, where sensors are associated to facilities, the contamination accident are associated to clients, and the cost of serving a client by a given facility represents the damage due to the accident if that sensor is the first one to detect the contamination, supposing that countermeasures are taken as soon as the contamination is detected. Since sensors are potentially located at any junction of the network and the number of contamination scenarios is quite large, the challenge is given by the enormous amount of information required to compute and store the cost coefficients, which incorporate the results of contaminant transport simulations for each scenario. Despite this difficulty, results are so good that this technology has already been put into practice at several water utilities. More recently, [22] exploited submodularity to compute near-optimal sensor placements, and solved the sensor location problem on a drinking water distribution system of more than 21,000 nodes. The key observation is that the function which computes the amount of population protected from consuming contaminated water by early detection provided by a given set of sensor locations is submodular.

Regarding the most suitable subset of devices to be activated, given the location of the first alerted sensor, the hydraulic engineering literature provides several approaches: both [2] and [19] propose a multi-objective approach minimizing the number n of operated devices as well as the impact on the population.

Regarding the actual schedule of devices activations, which is the subject of this study, this issue has only been partially addressed in the hydraulic engineering literature. In [6] the authors employ a genetic algorithm to determine the optimal flushing and valving operations in order to minimize the total network contaminant concentration fol-

lowing sensor detection. However, as well as [2], they suppose to activate all the selected devices instantaneously and simultaneously. More realistically, [19] 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.

The feasibility constraints of RCP coincide with those of a well known combinatorial optimization problem, i.e., the multiple Traveling Salesman Problem (see [7] for a recent review), as already mentioned. In the $mTSP$, m salesmen visit the nodes of a graph minimizing total traveled distance. Nodes must be assigned to salesmen and ordered within each salesman. The $mTSP$ literature devoted to the use of genetic algorithms provides useful suggestions. The lack of an analytical representation of tractable size for the RCP objective function leaves no room for mathematical programming based approaches, which could have taken advantage of the several studies on the polyhedral structure of the $mTSP$. Nevertheless, as this paper shows, the mathematical structure of the feasible region of RCP can be exploited in the framework of a meta heuristic method as a tool to restore or even impose solution feasibility, yielding hybrid solution approaches.

Differences between RCP and $mTSP$ concern the objective function: in the $mTSP$ it is easily computed, being the sum of traveled distances, while 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.

Since a good encoding should reflect the features characterizing the cost of the corresponding solution, the choice for an encoding is biased by the specific objective function to be optimized. As the solution cost in the $mTSP$ depends on selected arcs, which models pairs of successive nodes in the route, most GA encodings emphasize the information concerning the nodes sequencing, which is to say, the relative position of the nodes is more important than their absolute position in the sequence.

The two most common encodings are the One Chromosome and the Two Chromosomes encodings. The first one reports the sequences of nodes visited by each salesman separated by a symbol, say -1, different from any node identifier [28]. The second one is made of two vectors, the first one is a node permutation and the second one reports a salesman identifier for each node [23]. Both representations are heavily affected by redundancy. [9] proposes a new representation based on the so called Two Part Chromosome, which has a much lower redundancy. A computational study supports the evidence that redundancy hampers the search, since the same solution is visited several times as it corresponds to multiple individuals. In the experiments, ordered crossover was used for the first two encodings and for the first part of the Two Part Chromosome, while a single point asexual crossover method was used for its second part. However, a sort of local search was made concerning this second part, by trying several alternatives. The Two Part Chromosome based genetic algorithm produced statistically significantly better results. The Two Part Chromosome was also successfully applied in solving a scheduling problem in the newspaper industry [8]. More recently, [10] experimentally analyzes different crossover and mutation operators for the Two Part Chromosome on few instances of the TSPLIB.

The RCP feasibility structure is common to another well known family of sequencing problems in combinatorial optimization, i.e., the parallel machine scheduling problem with sequence dependent set up times ($PMSP$). In $PMSP$, each team is seen as a machine and each task as a job to be performed by a single machine, while sequence dependent set up times model traveling times. Therefore, also $PMSP$ requires partitioning and ordering decisions. Other than in $mTSP$, though, in $PMSP$ the focus is on the execution time of the jobs since the solution value usually depends on the schedule $t^{\mathcal{F}}$. For example, commonly used objective functions are the makespan, the total weighted completion time, the total weighted earliness or tardiness with respect to the job due date. Despite of the relevance of the execution time, this information is not directly encoded into the chromosomes of the genetic algorithms that have been proposed to tackle $PMSP$s. In general, the representations which are the most robust with respect to feasibility are the most exposed to redundancy, so that two chromosomes encoding similar solutions may be quite different, thus making it difficult for an

offspring to inherit their features. When cross over is performed directly on the solution, though, ad hoc repair operators are usually required to restore feasibility. $PMSP$ provides the perfect example. Due to the feasibility constraints that limit the values of $t^{\mathcal{F}}$, often the encoding models only the partitioning decisions while the schedule on each machine is obtained by applying some dispatching rules. [13], for example, uses a genetic algorithm to assign jobs to machines while machine scheduling on the individual machine is performed according to a greedy criterion. In [18] a random keys representation is used in a hybrid GA for the Job Shop Scheduling problem: schedules are constructed using a priority rule in which the priorities are defined by the GA, and a local search heuristic is applied to improve the solution. [27] tackles the parallel machine scheduling problem with earliness and tardiness penalties, where a set of independent jobs with sequence-dependent set up times and distinct due dates must be scheduled on a set of parallel machines in a non-preemptive fashion such that the sum of the weighted earliness and tardiness values of all jobs is minimized. Two genetic algorithms are proposed. The first one is based on the classical, sequence based, 2 chromosome representation, while idle times are handled by dispatching rules. Rules include classical ones, such as the *non delay* rule which schedules each job at its earliest start time, as well as original ones, such as the *forward pass* and *backward-forward pass* which aim at completing jobs at their due dates. The second one adds a third chromosome to the 2 chromosome representation, which indicates if a job must be scheduled at its ready time or at its best start time. In [30], multiple teams of photographers must be scheduled to serve a large number of schools. The photographers' schedules are to be optimized so that time constraints are satisfied and each team is able to visit at least two schools daily. The objective is to minimize total traveled distance and duration. An $mTSP$ model is used to formalize the problem which is then solved by a genetic algorithm that uses a sequence based encoding. In [11], a hybrid genetic algorithm is devised to solve a parallel machine scheduling problem to minimize the maximum weighted lateness. The genetic algorithm is used to evolve the job partition among the machines as well as the job sequence within each machine, while a local optimizer is used to improve the job order on each single machine. Even when scheduling on a single machine and despite of time based objective functions, permutation based encodings are preferred to time based encodings, as in [26] which solves a scheduling problem with set up times minimizing total tardiness.

Many other references can be found in [3], that provides a recent review of scheduling problems with set up times and costs. As far as we know, however, no encodings use the scheduling time directly in the chromosome.

4. A Genetic Algorithm Framework for the Scheduling of Devices Activation

A genetic algorithm (GA) draws inspiration from the mechanism of species evolution; each solution of a problem is encoded in some data structure, the *chromosome*. An initial population of individuals, each represented by a chromosome, is generated and then recombined through operators, most important the *crossover* operator. A crossover takes as input two chromosomes and generates a new individual that will be inserted into the new, evolved population. The candidate parents are often selected based on their *fitness*: a measure of the solution quality. Defining a GA, thus, 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 the limited computing time. Therefore, our termination condition is a fixed number of invocations to the hydraulic simulator.

The hydraulic and quality simulations were performed through EPANET [25], an open-source water distribution system modeling software package, developed by the U.S. Environmental Protection Agency (EPA). EPANET performs extended-period simulation of hydraulic and water-quality behavior within pressurized pipe networks, reproducing the movement of drinking-water constituents within distribution systems over time. EPANET also supports the simulation of spatially and temporally varying water demand, therefore it provides a realistic picture of user consumption. EPANET is able to describe the propagation of a contaminant through the network and

at which concentration, for each time slot and location, such contaminant is consumed during the day. In order to provide realistic results, the unit time slot of the discrete time simulation must be kept in the range of few minutes. At the same time, thought, the contaminant may take several hours to reduce its concentration below the danger threshold, and thus the simulation must encompass a large number of time slots. EPANET first performs a simulation of the flow in the hydraulic network over time with respect to the current schedule. Only afterwards, it computes how the contaminant spreads over the network and at which concentration and at what instant is present at each user demand node. Therefore, the volume of consumed contaminated water can be known only at the end of the simulation, so that the simulation process can not be interrupted before its end, for example as soon as the objective function value has reached a given threshold. Since each call to the hydraulic simulator is time demanding, we store the input/output data of each call in a sort of caching mechanism. 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.

In particular, the initial population is generated as follows. Each gene is created by picking at random a device identifier, according to a uniform distribution. If the selected device already appears in the chromosome, it is discarded and the process is repeated. Then, a team identifier is randomly selected, and it is assigned to the device. It is guaranteed that at least one device is assigned to each team. For the variable speed case, a third vector of n is created by sampling according to a uniform distribution the discrete set $\{0, \cdots, U\}$ of feasible pause durations. As it will be clear in the following, this representation is based on the Two Chromosome encoding; the translation of the initial population into the other encodings is straightforward.

In the next three sections we will recall two well known encodings taken from the literature on $mTSP$, and introduce a new one, targeting the specific features of our problem. A further variant for the ad hoc encoding will be also discussed. All these encodings will be described in case of constant traveling speed as well as variable traveling speed.

5. 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 $mTSP$.

The first encoding we consider is called the *two chromosome technique* by [9], and we will use the acronym $2C$ in the rest of the paper. The chromosome consists of two rows: the first represents the sequence of the devices to be activated, the second the identifier of the team that operates the corresponding device. For example, in the chromosome:

$$\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 3&4&1&2&8&5&7&6\\\hline 1&2&1&3&2&3&2&2\\\hline\end{array}}^{C_{dev}}\quad C_{team} \tag{2}$$

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 encoding, as all those based on permutations, is affected by redundancy which can slow down the convergence of the genetic algorithm [9]. For example, if we permute columns 1 and 2 in Eq. (2), we obtain exactly the same scheduling of the teams, but with a different representation. In fact, the first row of the $2C$ encoding gives a total order on the nodes, but in the scheduling the order of the nodes is actually significant only within each route. Ideally, the order of nodes should only be a partial order, since devices operated by different teams cannot be compared. The size of the solution space of this representation is $n!m^n$ [9].

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* [17]. Given two parents, f and m, and an 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 (3), we take $i = 4$ 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.

$$f = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 6&4&1&2&8&5&7&3\\\hline 1&2&1&3&2&1&2&2\\\hline\end{array}$$

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

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

The idea behind this operator is the following. The aim of a good crossover operator is having each 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 (3): 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 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 individuals, 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 offsprings 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.

5.1. Allowing for Variable Speed in 2C

As already mentioned, in RCP introducing *delays* in the schedule may improve the objective function value. To this purpose, the $2C$ encoding can be extended with a new vector C_{pause}, 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*. The resulting encoding consists of three chromosomes, but, for the sake of clarity, we call it "*Two Chromosomes encoding with Variable Speed*" ($2C^{VS}$).

It is worth pointing out that certain pause values might modify the order of the activations given by the first two chromosomes. For instance, considering the previous individual (2), for an opportune travelling time matrix τ we may obtain that the team 1, moving at *Constant Speed* (CS), activates the related sequence in $t_3^{CS} = \tau_{d3} = 7$ and $t_1^{CS} = \tau_{d3} + \tau_{31} = 9$; while the team 3 activates its sequence in $t_2^{CS} = \tau_{d2} = 5$ and $t_5^{CS} = \tau_{d2} + \tau_{25} = 8$. According with the resulting activation times, the total order of activations of the devices assigned to such teams is $\{2, 3, 5, 1\}$. Instead, the following possible representation of (2) in the space of $2C^{VS}$:

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

yields the activation times $t_3^{VS} = \tau_{d3} + 1 = 8$, $t_1^{VS} = \tau_{d3} + 1 + \tau_{31} + 5 = 15$, $t_2^{VS} = \tau_{d2} + 5 = 10$, and $t_5^{VS} = \tau_{d2} + 5 + \tau_{25} + 2 = 17$. In this way, the total order of activations of the concerned devices is changed in $\{3, 2, 1, 5\}$.

Regarding the crossover operator associated to the $2C^{VS}$ encoding, we extended the one-point ordered crossover in order to handle C_{pause} in the same way as the other two chromosomes.

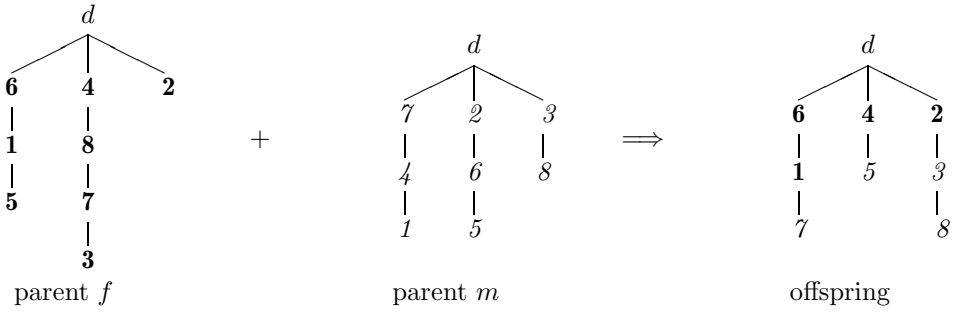
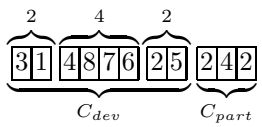


Fig. 1. An example of the tree representation of the two parents and the offspring obtained by the crossover

6. A Genetic Algorithm based on the Two-part Chromosome

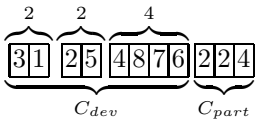
In [9] a so called *two-part chromosome* ($2P$) is proposed, with lower redundancy with respect to the other encodings so far proposed in the literature. The permutation part of the chromosome, C_{dev}, made of n integers as usual, is followed by a second part, C_{part}, being a string of m integers 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 (2) would become (5).



(5)

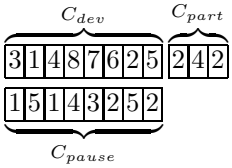
This way, the size of the representation space is lowered to the order of $n!\binom{n-1}{m-1}$. As in [9], we adopt the above mentioned one-point ordered crossover for the first chromosome part C_{dev}, and a single point asexual crossover (a random rotation) for the second one C_{part}. Both are closed with respect to this encoding and yield feasible solutions.

While the $2P$ 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. For example, the same solution of Eq. (5) could also be represented in $2P$ as follows:



6.1. Allowing for Variable Speed in $2P$

As well as in the $2C$ encoding, a straightforward extension of $2P$ for variable speeds, called $2P^{VS}$, can be achieved by defining the C_{pause} chromosome. In this way, the representation of (4) would become the folloing:



(6)

Also for the $2P^{VS}$, we provided a suitable extension of the one-point ordered crossover, in which the elements C_{dev_i} and C_{pause_i} are inherited jointly.

Finally, in $2C^{CS}$ (i.e., the basic $2C$), $2C^{VS}$, $2P^{CS}$ and $2P^{VS}$ GAs, we adopt the same mutation operator, i.e., swapping two columns of the chromosomes (i.e., C_{dev}, C_{team} for $2C$ encodings, and C_{pause} for VS encodings), and it is 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.

7. 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, their crossover operators do not allow to directly propagate the activation time of a device to

the next generation. Unluckily, the activation time 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 binary crossover operator (BX), which selects genes from the two parents based on a randomly generated binary mask. A time-based GA based on BX 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. For example, if the activation times of the two parents are as follows

$$f = \boxed{3}\boxed{4}\boxed{3}\boxed{1}\boxed{8}\boxed{5}\boxed{7}\boxed{6} \qquad m = \boxed{1}\boxed{8}\boxed{1}\boxed{3}\boxed{6}\boxed{3}\boxed{2}\boxed{2}$$

and the bit mask selects the times in odd positions from f and the ones on even positions from m, the spawned child would be

$$c = \boxed{3}\boxed{8}\boxed{3}\boxed{3}\boxed{8}\boxed{3}\boxed{7}\boxed{2}$$

Notice that the child c has four devices that must be operated at time 3. This means that if we have tree teams, it is impossible to operate four devices exactly in the same instant, although it was possible for each of the parents.

In order to have only feasible schedules in the population, we apply a step to restore feasibility after the application of each genetic operator.

In the following, we hybridize in two ways the GA with a Mixed Integer Linear Programming (MILP) solver. In particular, we introduce a 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 BX crossover, and this approach will be denoted as BX *with a posteriori feasibility restore* ($BXPF$). Furthermore, we extend this idea and integrate the MILP model directly within the genetic operator, giving raise to a second approach denoted as $MILPX$.

7.1. 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 $t^{\mathcal{F}}$ 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:

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

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 $t^{\mathcal{F}} = [2, 1, 1, 3]$ 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 $t^{\mathcal{F}}$, building on those developed for the $mTSP$ [7] and routing problems in general, among which the following *2-index flow-based* formulation [29]. 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).
- $t^{\mathcal{F}}$ a vector of $n+1$ activation times; $t_i^{\mathcal{F}}$ is the time at which device i is activated, and $t_d^{\mathcal{F}}$ is the departure time from the depot d.
- δ a vector of n differences: it is defined as $\delta_i = t_i - t_i^{\mathcal{F}}$.

The input parameters are:

- t a vector of n ideal activation times.

$$\forall i \in \{1..n\} \qquad t_i^{\mathcal{F}} \geq \tau_{di} \tag{7}$$

$$\forall i \in \{1..n\} \qquad \delta_i = t_i - t_i^{\mathcal{F}} \tag{8}$$

$$t_d^{\mathcal{F}} = 0 \tag{9}$$

$$\sum_{i \in \{1..n\}} x_{di} = m \tag{10}$$

$$\forall i \in \{1..n\} \qquad \sum_{j \in \{1..n\} \cup d} x_{ij} = \sum_{h \in \{1..n\} \cup d} x_{hi} \tag{11}$$

$$\forall i \in \{1..n\} \qquad \sum_{j \in \{1..n\} \cup d} x_{ij} = 1 \tag{12}$$

$$\forall i \in \{1..n\} \qquad t_i^{\mathcal{F}} \leq M + x_{di}(\tau_{di} - M) \tag{13}$$

$$\forall i, j \in \{1..n\} \qquad t_i^{\mathcal{F}} + \tau_{ij} \leq t_j^{\mathcal{F}} + (1 - x_{ij})M + x_{ji}(\tau_{ij} + \tau_{ji} - M) \tag{14}$$

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: Constraint (7) says that device i can be activated no earlier than the time it takes to reach it from d.
Eq. (8) is the definition of δ.
Teams leave the depot at time 0 (Eq. (9)).
All m teams depart from the depot (Eq. (10)).
For each node i, the total number of teams arriving to i is equal to the number of teams leaving i.
Eq. (11) are the so called flow balance constraints.
All nodes except d are visited exactly once (Eq. (12)).
Constraint (13) is the linearization of the implication $x_{di} = 1 \Longrightarrow t_i^{\mathcal{F}} \leq \tau_{di}$, where M is a sufficiently large positive number so that, together with Constraint (7), it imposes that the starting time of the first devices must be equal to their traveling time from d. Constraint (14) links the activation times $t^{\mathcal{F}}$ to the ordering between devices given by matrix X; indeed, (14) linearises the implications:

$$x_{ij} = 1 \implies t_i^{\mathcal{F}} + \tau_{ij} \leq t_j^{\mathcal{F}}$$

$$x_{ij} = 1 \implies t_i^{\mathcal{F}} + \tau_{ij} \geq t_j^{\mathcal{F}}.$$

Eq. (14) 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*. The objective function associated to problem (7-14) is the minimization of the L_1 distance between $t^{\mathcal{F}}$ and t, namely

$$min||\delta||_1 \quad = \quad min\left(\sum_{i \in \{1..n\}} |\delta_i|\right) \tag{15}$$

To linearize this function, we introduce new unknowns δ^+ that represent the absolute value of δ, and minimize their sum. We call this problem the *Feasibility Restoring Problem* (FRP).

7.1.1. Problem Complexity

Theorem 1. *The* Feasibility Restoring Problem (FRP)*, aiming at finding the feasible vector $t^{\mathcal{F}}$ of activation times that is closest, according to norm L_1, to the ideal vector t, is NP-Hard.*

Proof. FRP can be reformulated in the framework of machine scheduling problems. Consider m parallel identical machines, a set of n jobs to be executed, and a distinct due date t_i for each job $i = 1, \cdots, n$. Each job i has a duration d_i and there are sequence-dependent set up times, i.e., a set up time s_{ij} must be spent if job i is executed right before job j on the same machine. Set up times to initialize the machine before the first job are also present. Each job must be assigned to exactly one machine. Each machine can execute a job at a time, and preemption is not allowed. Therefore, jobs on the same machine are totally ordered. The target is to minimize the sum of earliness and tardiness penalties with respect to the due dates. This problem is NP-Hard, since it is a generalization of

the scheduling of independent jobs with a common due date on a single machine [14].

The FRP can be stated in terms of the machine scheduling problem introduced above as follows: each job corresponds to a device to be activated, job durations are null (which comes at no loss of generality since positive durations can be included in the set up times), due dates correspond to the ideal activation times t, earliness and tardiness penalties for each job are unitary, set up times $\{s_{ij}\}$ are the traveling times on the network $\{\tau_{ij}\}$, and initialization set up times correspond to the traveling times from the depot. Regarding the objective function, the distance between the two vectors $t^{\mathcal{F}}$ and t according to norm L_1 is in fact the sum of the absolute values of the differences, which corresponds to the sum of earliness and tardiness of the scheduling time of the jobs with respect to their due date. This fact, beside the need for linearization, motivates the choice of the L_1 norm to evaluate the distance between $t^{\mathcal{F}}$ and t.

Summarizing, since FRP can be restated as an NP-Hard machine scheduling problem, it follows that FRP is NP-Hard as well. □

Since there is no known polynomial algorithm for the exact solution of FRP, it is reasonable to adopt an exponential worst case complexity approach, such as formalizing FRP by the MILP model provided in (7-14) and solving this model by a MILP solver.

7.1.2. A Variable Speed Variant for $BXPF$

The time encoding allows us to encompass the variable speed variant of $BXPF$ without the need of adding a vector C_{pause} to the chromosome, as it was necessary for both the Two Chromosome and the Two Parts Chromosome in order to encode the pause before the activation of each device. Indeed, the time encoding already possesses all the information required. What is affected, though, is the feasible region of the time vectors, since now a chromosome is feasible as long as there is a solution to the associated $mTSP$ such that the activation time of each device coded in the gene is *greater than or equal to* the activation time of the preceding device plus the traveling time. It follows that the MILP model for the FRP must be modified to take into account this broader feasible region. Recall that U denotes the maximum pause allowed. Constraints (13) and (14) are now modified as follows.

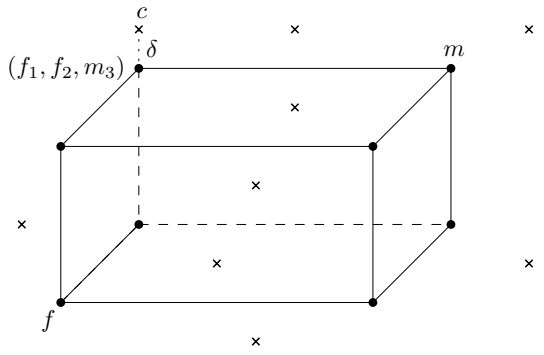


Fig. 2. Graphic representation of the crossover in a 3D space. Crosses represent feasible points, m and f are the mating individuals; c is the closest feasible point (at distance δ) to a vertex of the parallelepiped.

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

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

7.2. 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 rise 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 its parents as much as possible among all their feasible children.

More precisely, given the chromosomes of the mating individuals $f \equiv (f_1, \ldots, f_n)$ and $m \equiv (m_1, \ldots, m_n)$, 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 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 (7-14), by introducing a vector of unknowns w to range on the vertices 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 (8) of the displacement δ becomes:

$$\forall i \in \{1..n\} \qquad \delta_i = f_i w_i + m_i(1-w_i) - t_i^{\mathcal{F}}. \quad (16)$$

In order to avoid the offspring to be a clone of one of the parents, an additional set of constraints and a new family of variables are introduced. Variables model the distance between the offspring and each of the parents; the constraints require such distances to be greater than a positive threshold. Note that the same condition can not be guaranteed by simply bounding the number of coordinates coming from each parent, i.e., $\sum_{i\in\{1..n\}} w_i$ and $n - \sum_{i\in\{1..n\}} w_i$, to be at least 1 and less than n. For example, this condition has no effect when parents have one or more genes with the same value: suppose that $f_1 = m_1$, the solver could select a child identical to parent m by selecting $w_1 = 1$; in this way $\sum_i w_i = 1$ although the child c is identical to one parent. Near the end of the search, it is not uncommon to have parents with equal value of one coordinate.

$MILPX$ and $BXPF$ have different characteristics. Since the two parents are feasible vertices of the hyper-parallelepiped, $MILPX$ may become quite conservative as the search proceeds. In fact, $MILPX$ tends to reproduce entire patterns of the parents, such as the scheduling of single teams. In this way it may soon reduce the diversity of the population, even though it succeeds in transmitting meaningful information. On the contrary, $BXPF$ may disrupt the scheduling patterns of the parents but it may help to divert the search away from local optima.

Both $MILPX$ and $BXPF$ select a reference vertex of the hyper-parallelepiped, to which the feasible offspring has to be as close as possible. $MILPX$ and $BXPF$ can be seen as the two extremes concerning the use of randomness in the choice of the reference vertex. In $BXPF$, it is a pure random choice, implemented by the binary mask of BX. On the contrary, in $MILPX$ the choice is fully deterministic and it is driven by feasibility. In fact, $MILPX$ selects as the reference vertex the vertex whose distance from the closest feasible point is minimum. There is no way to tell in advance which crossover operator is the best one. Both crossovers can be used within a hybrid GA, each one being used according with a given probability. Different values of such a probability are evaluated experimentally, as reported in Section 8.

Summing up, for the GA based on activation times, we propose two crossovers, $BXPF$ 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 to the current population (a clone), and consists of swapping the activation time of two devices, restoring feasibility if necessary.

8. Computational Results

We applied the presented GAs to the water distribution network of Ferrara, Italy, population 130,000. A sketch of the network is reported in Figure 3. Topology network data are sensitive information therefore can not be distributed.

In the current experiments, the road network on which the teams move along, coincides with the water distribution network. We claim that this occurs at no loss of generality, regarding the assessment of the efficiency of the solution approach, since there is no relationship between how the teams move on the road network and how the contaminant spreads over the hydraulic network. What matters for a significant experimental study is that the travel time between devices on the road network is not negligible, so that the order in which devices are operated affects devices activation times of a quantity sufficiently high to impact contaminant spreading. This condition is satisfied by our instances.

A previous work on the same network [19] 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

Fig. 3. Picture of the hydraulic distribution network of Ferrara

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 [19], 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 [19], 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. We considerably extended the experimental campaign of [16], which had been carried out on 5 scenarios equally spread with respect 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 (7), (9-14), and $U = 0$, minimizing the maximum among the devices activation times $\{t_i^{\mathcal{F}}, i \in 1..n\}$, which is also called the *makespan*. In this study we selected 20 scenario, including the previous 5, selected according to the same criterion.

With respect to [16], we improved the quality of the MIP solver used to tackle the optimization problems in the Hybrid GAs and to compute the minimum makespan schedule. CBC COIN-OR [12] is the open-source code MILP Solver used in [16], while in this study we adopted a commercial MILP Solver, namely Gurobi [20]. Pros and cons are discussed further on.

As mentioned, EPANET is the hydraulic simulator used in this study, an open-source software developed by the U.S. Environmental Protection Agency (EPA) which has become a standard tool in the hydraulic engineering literature [25]. Each simulation requires on average about 5 seconds. This time is almost constant, since it is mainly determined by the size of the time steps and by the length of the simulation period. Each call to the MILP solver is, on average, in the order of a few tens of milliseconds, so that computing time for solving one instance is basically proportional to the number of EPANET calls. Note that this number corresponds to the number of different solutions inspected during the search, since the cache memory mechanism spares simulation time in case of solutions already evaluated. According to the needs of the local water utility, the total computing time for solving one instance should not exceed one hour. Therefore, a reasonable choice is to set a cutoff of 500 invocations to the hydraulic simulator. The average computational time of each GA is approximately 5×500 seconds, and variance is negligible.

Other parameters are the population size $N_{pop} = 20$, and the team number $m = 3$. The value of m was set by the managers of the utility company operating the Ferrara network. With these parameters, Gurobi's running time is negligible w.r.t. EPANET, as already mentioned. We set a time out limit of 1 second to ensure limited variability.

Overall, we ran 14 GAs. The first 10 belong to the time-based Hybrid GAs family (section 7) 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 $BXPF$ as the crossover operator at the current iteration. More specifically,

we tested five $MILPX$ probability values, namely $\{0, 25, 50, 75, 100\}\%$. We name them with respect to the percentage of $MILPX$, i.e., the variable speed variant of the hybrid GA where $MILPX$ is used with probability 0.25 and $BXPF$ has probability 0.75 is denoted as $H(25\%, VS)$. The other four GAs belong to the permutation-based family (sections 5 and 6), namely, $2C^{CS}$, $2C^{VS}$, $2P^{CS}$, and $2P^{VS}$ GAs. For each scenario, we run each GA 100 times. Each GA, at each run, shares the same initial population as the other GAs for the same variant (constant and variable speed), which is to say, there are 200 different initial populations, the first 100 are made of feasible solutions for the variable speed case, while the remaining 100 are feasible also for the constant speed case.

The experimental campaign aims at tuning the population size; providing computational evidence of the gain given by using time based encodings with respect to permutation based ones for solving this problem; deriving some indications concerning the best settings of the hybrid genetic algorithm, such as the percentage of $MILPX$ and $BXPF$ and the use of powerful MILP solvers; analyzing whether the devised solution approach is able to cope with the more challenging version of our problem, that is the variable speed variant.

Population size N_{pop} has been calibrated experimentally, by running on few scenarios algorithm $2C^{CS}$. All scenarios yielded similar results. We report the data for scenario A in Figure 4. The average over 100 runs of the best solution value at each EPANET call is reported, for population size in $\{20, 40, 60, 80, 100\}$. Larger populations allow to start the search from better quality solutions, but this advantage is not compensated by the low number of generations allowed by our stopping criterion, i.e., 500 invocations to the hydraulic simulator. Having to deal with this strong limitation, we suppose that our algorithms perform better if generating less offspring from an improving population rather than generating more offspring from the same population. Figure 4 suggests in fact to set $N_{pop} = 20$.

Despite the fact that in the genetic algorithm literature N_{pop} is usually much larger than 20, i.e., 100, it should be mentioned that, in some studies such as [26], calibration yielded much smaller values, such as 40.

The running time allowed to each algorithm is basically equal, and most important, is equal the

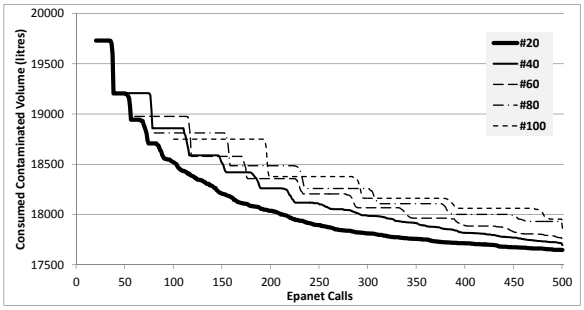


Fig. 4. Population size calibration, average on 100 runs of the best solution value of $2C^{CS}$ at each EPANET call, scenario A, 500 simulation calls

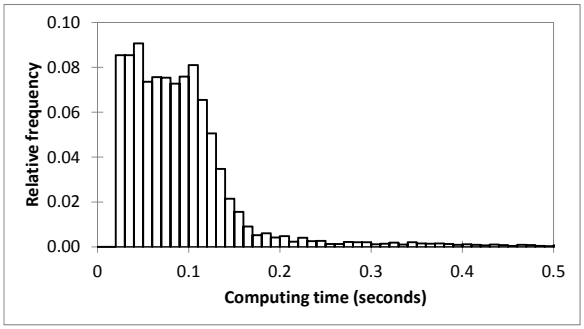


Fig. 5. MILP solver running times distribution over 9000 runs

number of solutions each algorithm inspects, i.e., 500. As mentioned, the MILP solver running time is negligible with respect to the simulation time, so that all the hybrid algorithms take, on average, the same amount of time as the others. Data reported in Figure 5 confirm this fact. The distribution of the Gurobi MILP solver running time on 9000 runs is depicted. The runs have been randomly sampled during several executions of our hybrid algorithms. Each bar shows the number of runs whose running time lies within the corresponding interval. It can be seen that the vast majority of the values lies below 0.12 seconds.

Let us discuss pros and cons of using high performance MILP solvers in our hybrid GAs. We switched from CBC to Gurobi in order to improve robustness with respect to running time and results. Both solvers were used with a time out limit set to 1 second, which CBC used to reach sometimes while Gurobi seldom reaches it. So far with the pros. Cons regards an increasing number of clones being generated by $MILPX$. We then

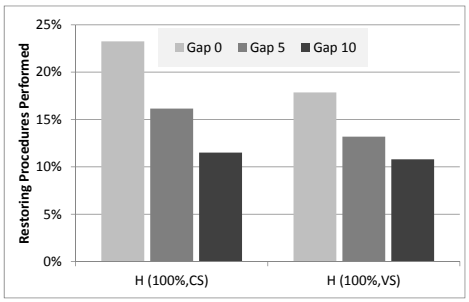


Fig. 6. Impact of tolerance gap on the number of feasibility restore after mutations.

analyze the data for the hybrid GA with 100% $MILPX$.

Recall that $MILPX$ tends to propagate parents feasible patterns, so that, when two parents are similar and population diversity is low, $MILPX$ tends to produce clones. While we forbid an offspring to be a clone of its parents, there is no way to forbid it to be equal to an individual in the current population. Mutation occurs when a clone is produced, which very often requires a feasibility restore, and then another call to the solver. Willing to keep the good features of Gurobi, we addressed this issue by allowing a positive tolerance in accepting suboptimal solutions, namely we accept a suboptimal solution with an absolute gap of 5 and 10 minutes. This trick sharply reduced the number of calls to the solver due to feasibility restore after mutation, in both settings, constant and variable speed. Figure 6 shows, for both cases, the percentage of calls to the MILP solver due to a restore following a mutation, computed with respect to the total number of calls; this number includes one call for each crossover operation, which determines the reference vertex and returns its closest feasible solution as the offspring. It can be seen that, in case of constant speed, an absolute tolerance of 10 halves, on the average, the percentage of calls devoted to feasibility restore, and a tolerance of 5 reduces it roughly by a quarter. This effect is evident, although less significant, also in case of variable speed. A possible explanation is the following. In case of null tolerance, the solver determines the optimal solution. Disregarding the case of multiple optima, there is a unique vertex which can be the reference vertex. The solver simply identifies it but does not take any decision. On the contrary, in case of a positive tolerance, more than one vertex can be the reference vertex, and which one will be selected by the solver depends on several features regarding the search strategies of the solver, which we can not control. Seen from the outside, then, it is a sort of random choice among the potential reference vertices. From this point of view, i.e., regarding the degree of randomness in the choice of the reference vertex, the $MILPX$ crossover with tolerance lies in between the pure $MILPX$ and $BXPF$.

Table 1
Average number of calls to the MILP solver, averaged on 5 scenarios for 100% $MILPX$

	Constant Speed		Variable Speed	
GAP	**MILPX**	**Restore**	**MILPX**	**Restore**
0	779	236	634	138
5	692	133	590	90
10	611	79	564	68

Table 1 reports the absolute number of MILP solver calls, averaged over 5 scenarios, for both constant and variable speed. It can be noticed that a positive tolerance not only reduces the number of calls due to mutations of clones, but it also reduces the number of calls due to the $MILPX$ crossover. In fact, after mutation, another clone could be obtained if the feasibility restoring process maps the mutated schedule to a solution already present in the population. In such a case, another crossover operation is performed, thus increasing the number of calls to the MILP solver. On the contrary, a wider choice is allowed by relaxing the optimality requirement regarding the distance from the infeasible mutated schedule, and chance may take to a new solution which is not part of the current population. Again, the effect is more evident for variable speed.

The different behavior in case of constant or variable speed may be due to the following reason. The variable speed feasible region is much wider that the constant speed one, and it is even locally compact. Therefore, also the set of the feasible values for the operation time of a device is much wider, and there are likely to be several feasible points in the neighborhood of each vertex, as well as several vertices which have a feasible point close enough to be selected. All these features may help to keep the search enough diversified and then reach better solutions.

As a conclusion, we run all the following experiments with the tolerance threshold equal to 10 for the $MILPX$ crossover.

Figure 7 provides an overall picture of the quality of the results of the different GAs, by examining how good each of them ranks over the 100 runs on all the 20 scenarios. In particular, we count how many times each GA ranks first, second, third, and within the first three positions, respectively, over a total of 100×20 runs.

Considering variable speed, one can say that both permutation based encodings are not competitive with respect to any time based encoding; considering constant speed, the same holds except for $H(100\%, CS)$ whose performance appears to be close to the one of $2C^{CS}$. This confirms our hypothesis that, in this problem, the information related to the operation time of a device is more relevant than the information related to the relative order of the devices operation times.

It can also be noted that the most performing encoding for the $mTSP$, i.e., $2P^{CS}$, performs worse than the classical $2C^{CS}$. The same holds for the variable speed variant. This may be due to the fact that both $2C^{CS}$ and $2C^{VS}$, used together with the one-point ordered crossover, are more likely than $2P^{VS}$ and $2P^{CS}$ to transmit information regarding the first devices to be operated by the teams, and such devices seem to have a major impact on the pattern according to which water flows.

It can be observed that it is worth facing the challenge of the variable speed variant. The most performing algorithm is indeed $H(25\%, VS)$. Moreover, any hybrid variable speed GAs improves on any permutation based GAs.

The impact of the relative presence of the two crossovers in the hybrid GAs varies depending on which speed variant is considered. While in the constant speed case the performance improves with the probability of the $BXPF$ crossover, in variable speed, the $MILPX$ crossover shows some degree of efficacy. This may be related to the observations we made regarding the fact that the variable speed variant of $MILPX$ is less affected by the phenomenon of clones.

In order to estimate the effectiveness of the algorithms we performed a significance test analysis. The choice of a statistical test should take into consideration the properties of the available data set.

Consider the values of the solutions returned by a GA after a given number of runs. This population does not follow any particular distribution. For each scenario, we have 100 different runs for each algorithm, and the initial populations are equal for each run, within the same speed variant. Therefore, in our case we can say that: i) for each scenario and for each speed variant we have a "*complete block design*" of experiments, i.e., each algorithm is tested over the whole set of initial populations. ii) the results related to the same speed configuration and the same scenario can be thought of as a "*paired*" dataset, because the i-th run of each GAstarts from the same initial population. Therefore, the data associated to a given scenario and a given speed configuration satisfies the conditions required to apply the *Friedman test*.

The Friedman test computes the significance level of rejecting the so called "null" hypothesis, i.e., that there are no different behaviours among the algorithms. If the resulting *p-value* is lower than an opportune confidence level α (commonly equal to 0.05), the null hypothesis can be rejected with a good confidence level.

Whenever the p-value of a Friedman test is low enough, one can perform the *Nemenyi post hoc analysis* in order to find out which pairs of algorithms have (significantly) different behaviours. The Nemenyi procedure consists of pairwise comparisons among the set of selected algorithms. However, although confidence levels in each single comparison can be low, the probability of having at least one error increases with the number of comparisons. In general, in order to ensure that a multiple comparisons test yields significant values, the standard confidence level can be made more tight by using an opportune correction method. In literature, the most known (and the most conservative) method is the *Bonferroni correction*, which defines the new confidence level as $\alpha^* = \alpha / Numberofcomparisons$).

The above mentioned tests have been carried out by way of XLSTAT, which is a commercial plug in for Microsoft Excel [1]. A useful reference textbook for nonparametric statistical analysis is [21].

All the Friedman tests performed on the available complete block design datasets return a p-value lower than 0.05, often under 0.0001. This means that for any scenario and speed variant there are algorithms that perform better than someone else. To analyse if there are recurrent statistical significant comparisons of GAs over the whole set of scenario, we first performed the Ne-

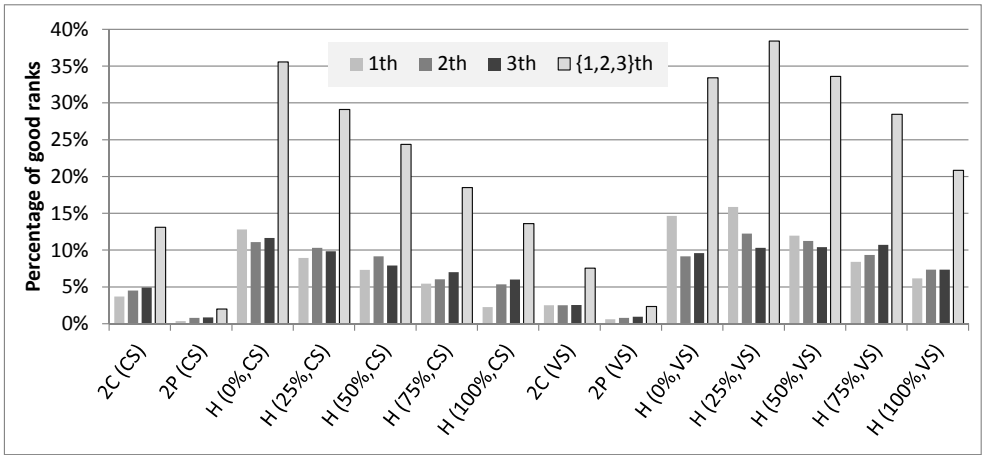


Fig. 7. Rankings 100 × 20 runs

menyi test procedure on each consistent dataset, and then, for each possibile comparison, we computed the number of times in which a comparison is statistically significant (given the opportune correction) over the 20 scenarios.

The first aim is to identify if, in general, the Hybrid GAoutperforms the other based on sequences, both for the constant and the variable speed. The percentage of significant comparisons between each $GA$with $2C$ and $2P$ GAs are shown in Table 2 for constant speed variants, and in Table 3 for variable speed, whose values are depicted respectively in Figure 10 and 11. It is very clear that both the $2P^{CS}$ and $2P^{VS}$ are outperformed by the Hybrid GA on about the whole set of scenarios, and, symmetrically (but not obvious), the two speed variants of $2P$ never were significantly better than the others. The Hybrid GA outperforms the $2C^{VS}$ for all the $MILPX$ configurations between 80% and the 100% of scenarios. But for the constant speed variants the chart in Figure 10 shows that the Hybrid GA loses effectiveness increasing the $MILPX$ selection chance.

In order to delve into the effectivness of $MILPX$ an speed variants configuration of the Hybrid GA, we performed the Nemenyi procedure only over the datasets related to the Hybrid algorithm. Tables 4 and 5 report, as well as in the previous tables, the percentage of significant comparisons over the set of 20 scenarios of the several tested configurations, and an overview is given by Figure 8 and 9. Even here, a lower influence of the fast convergence for higher $MILPX$ selection chances

Table 2

Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Constant Speed GA variants and configurations, only the values of the comparisons with $2C^{CS}$ and $2P^{CS}$ are shown ($\alpha^* = 0.05/21$)

	$\mathbf{2C^{CS}}$	$\mathbf{2P^{CS}}$
$\mathbf{2C^{CS}}$	-	90%
$\mathbf{2P^{CS}}$	0%	-
H (CS,0%)	95%	100%
H (CS,25%)	70%	100%
H (CS,50%)	50%	100%
H (CS,75%)	35%	100%
H (CS,100%)	15%	90%

Table 3

Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Variable Speed GA variants and configurations, only the values of the comparisons with $2C^{VS}$ and $2P^{VS}$ are shown ($\alpha^* = 0.05/21$)

	$\mathbf{2C^{VS}}$	$\mathbf{2P^{VS}}$
$\mathbf{2C^{VS}}$	-	20%
$\mathbf{2P^{VS}}$	0%	-
H (VS,0%)	100%	100%
H (VS,25%)	100%	100%
H (VS,50%)	95%	100%
H (VS,75%)	85%	100%
H (VS,100%)	15%	90%

can be noticed for the variable speed cosnfigurations; in fact, the most significant configuration is $H(25\%, VS)$, which outperforms $H(100\%, VS)$ only for 10 scenarios and $H(0\%, VS)$ for 6 scenar-

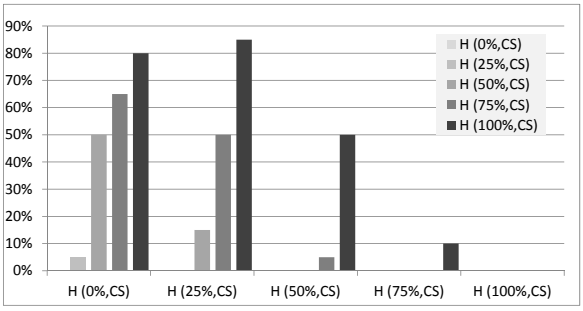


Fig. 8. Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Constant Speed Hybrid GA configurations from Table 4 ($\alpha^* = 0.05/10 = 0.005$)

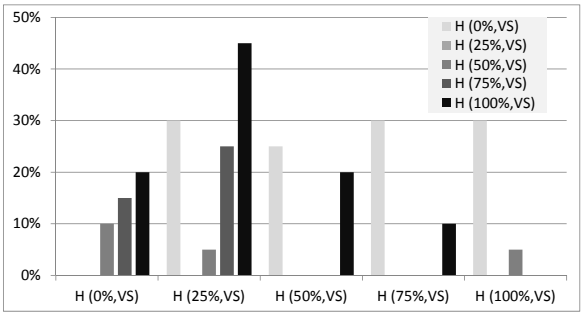


Fig. 9. Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Variable Speed Hybrid GA configurations from Table 5 ($\alpha^* = 0.05/10 = 0.005$)

ios (see Figure 9). Instead, the chart in Figure 8 shows that both $H(0\%, CS)$ and $H(25\%, CS)$ outperform $MILPX$ configurations greater than 50% between 10 and 17 scenarios.

The following boxplots provide a schematic view of the performance of the 14 GAs for each of the 20 scenarios, for all the 100 runs. The specific scenario affects the performance of all the GAs. In fact, the variance of the results of each GA varies on the different scenarios according to same pattern; for example, all GAs have a high variance on scenario P and a much lower one on scenario T.

We noticed that the GAs achieve very different levels of contaminated consumed water on different scenarios, whereas their relative performance, i.e., how good a GA is with respect to another one, does not vary considerably according to the scenario. Since the quantity of contaminant injected is the same for each scenario, we wonder whether the set of the n devices to be operated is targeted

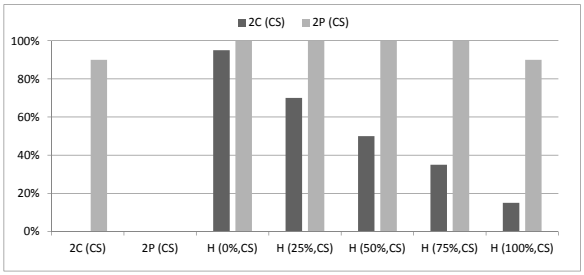


Fig. 10. Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Constant Speed GA variants and configurations from Table 2 ($\alpha^* = 0.05/21 = 0.0024$)

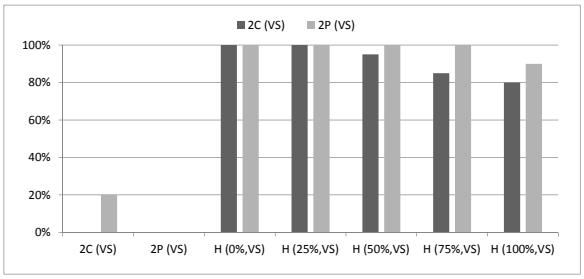


Fig. 11. Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Variable Speed GA variants and configurations from Tab. 3 ($\alpha^* = 0.05/21 = 0.0024$)

equally well for each scenario. Moreover, we can observe that in the worse scenarios, namely I and P, the variable speed variant hybrid GAs are not competitive with respect to the ones at constant speed. We argue that this might be due to the choice of the devices to be operated. The resulting shape of the objective function may thus be quite rough, and this may be a too difficult obstacle to deal with, beside the wider feasible region, with such a limited sized population.

Table 4

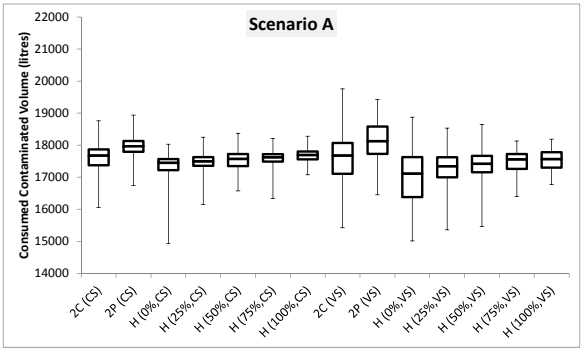
Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Constant Speed Hybrid GA configurations ($\alpha^* = 0.05/10 = 0.005$)

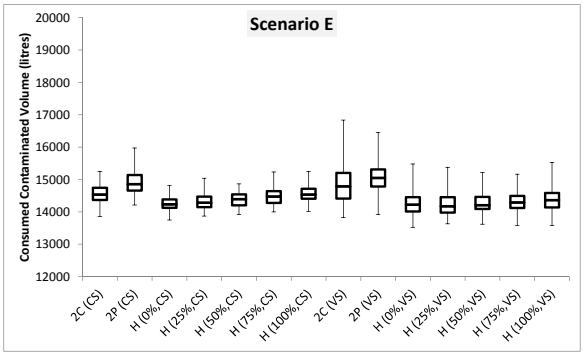
	H (CS,0%)	**H (CS,0%)**	**H (CS,0%)**	**H (CS,0%)**	**H (CS,0%)**
H (CS,0%)	-	5%	50%	65%	80%
H (CS,25%)	0%	-	15%	50%	85%
H (CS,50%)	0%	0%	-	5%	50%
H (CS,75%)	0%	0%	0%	-	10%
H (CS,100%)	0%	0%	0%	0%	-

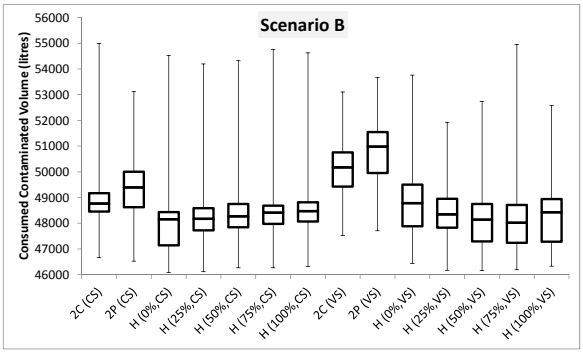
Table 5

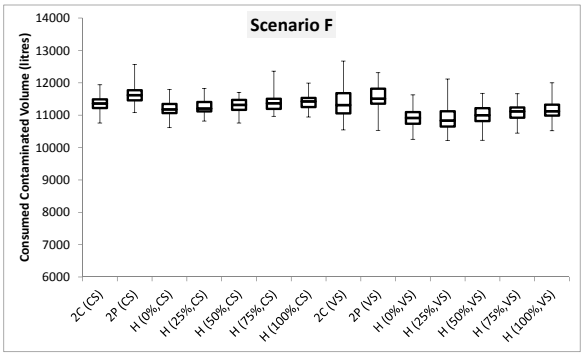
Nemenyi Post Hoc Analisys: percentage of significant comparisons within the 20 scenarios of the Variable Speed Hybrid GA configurations ($\alpha^* = 0.05/10 = 0.005$)

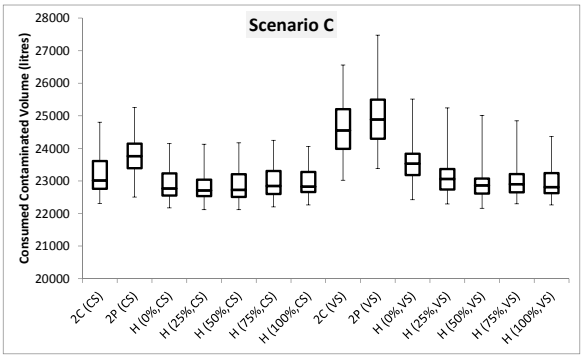
	H (VS,0%)	**H (VS,0%)**	**H (VS,0%)**	**H (VS,0%)**	**H (VS,0%)**
H (VS,0%)	-	0%	10%	15%	20%
H (VS,25%)	30%	-	5%	25%	45%
H (VS,50%)	25%	0%	-	0%	20%
H (VS,75%)	30%	0%	0%	-	10%
H (VS,100%)	30%	0%	5%	0%	-

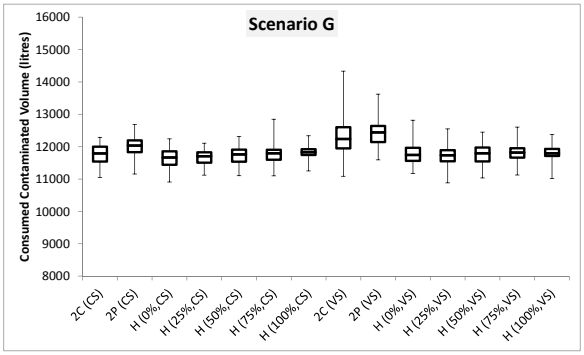

Scenario A
Consumed Contaminated Volume (litres)
22000
21000
20000
19000
18000
17000
16000
15000
14000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

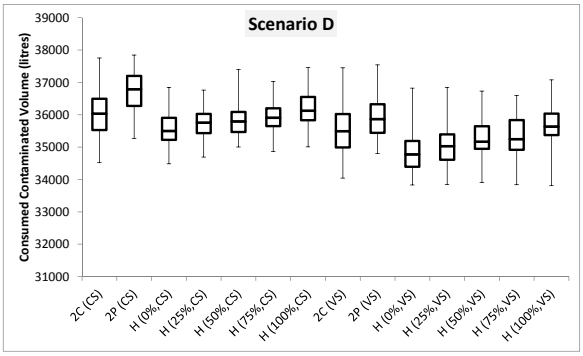

Scenario E
Consumed Contaminated Volume (litres)
20000
19000
18000
17000
16000
15000
14000
13000
12000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

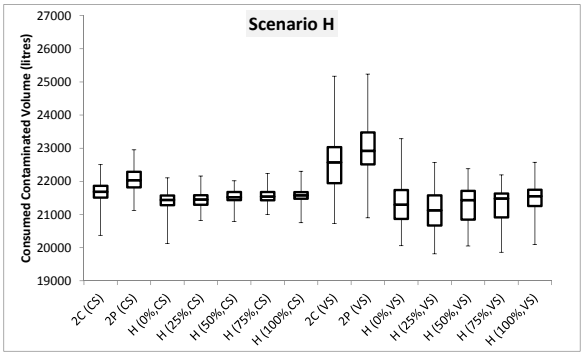

Scenario B
Consumed Contaminated Volume (litres)
56000
55000
54000
53000
52000
51000
50000
49000
48000
47000
46000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

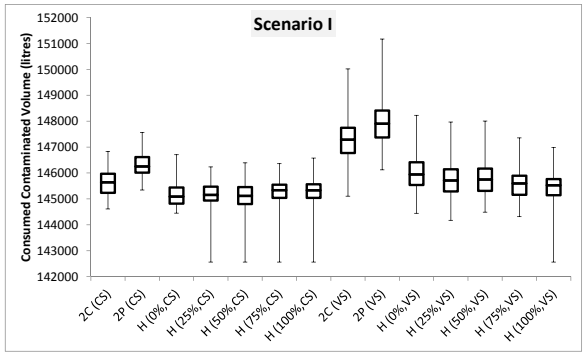

Scenario F
Consumed Contaminated Volume (litres)
14000
13000
12000
11000
10000
9000
8000
7000
6000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

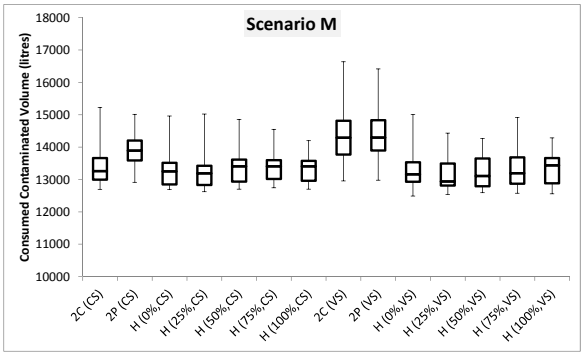

Scenario C
Consumed Contaminated Volume (litres)
28000
27000
26000
25000
24000
23000
22000
21000
20000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

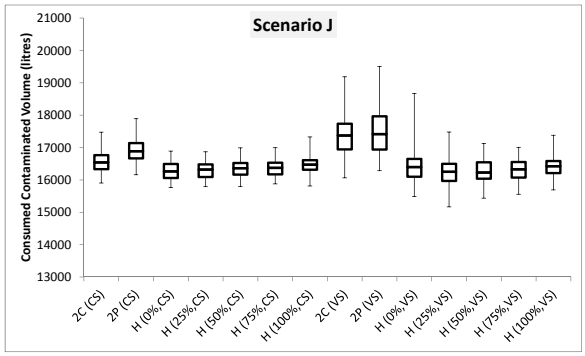

Scenario G
Consumed Contaminated Volume (litres)
16000
15000
14000
13000
12000
11000
10000
9000
8000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

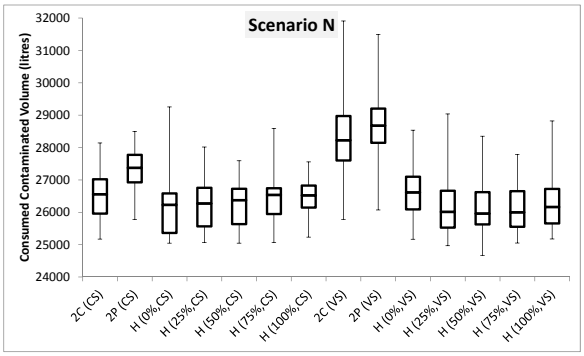

Scenario D
Consumed Contaminated Volume (litres)
39000
38000
37000
36000
35000
34000
33000
32000
31000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

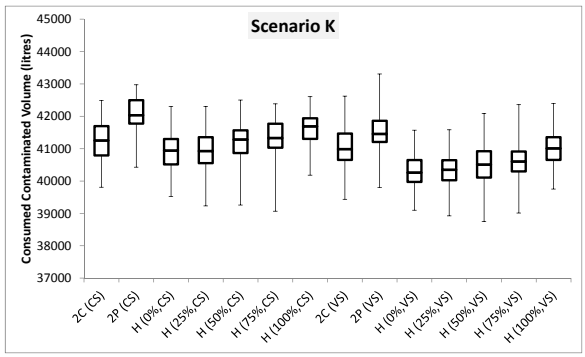

Scenario H
Consumed Contaminated Volume (litres)
27000
26000
25000
24000
23000
22000
21000
20000
19000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

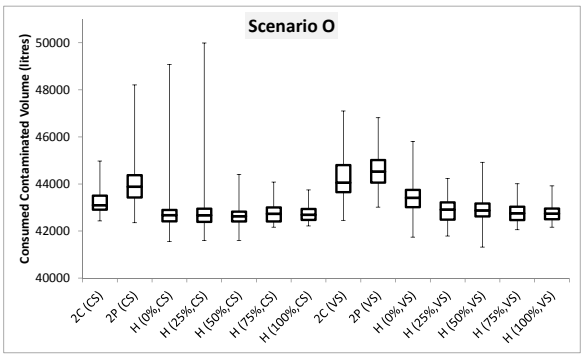

Scenario I
Consumed Contaminated Volume (litres)
152000
151000
150000
149000
148000
147000
146000
145000
144000
143000
142000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

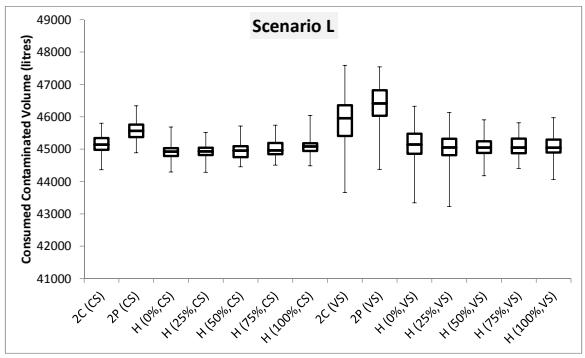

Scenario M
Consumed Contaminated Volume (litres)
18000
17000
16000
15000
14000
13000
12000
11000
10000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

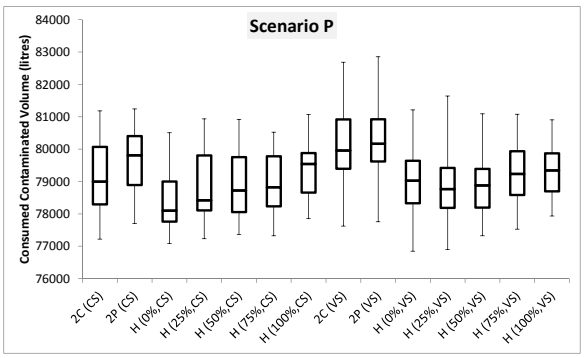

Scenario J
Consumed Contaminated Volume (litres)
21000
20000
19000
18000
17000
16000
15000
14000
13000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

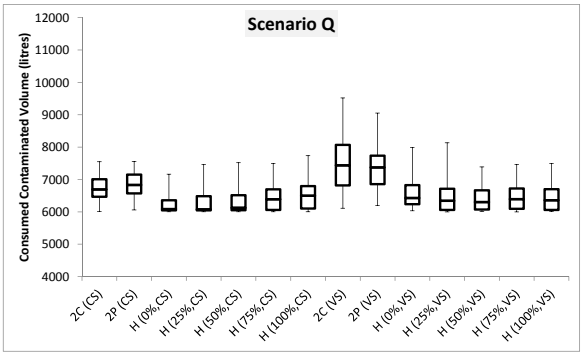

Scenario N
Consumed Contaminated Volume (litres)
32000
31000
30000
29000
28000
27000
26000
25000
24000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

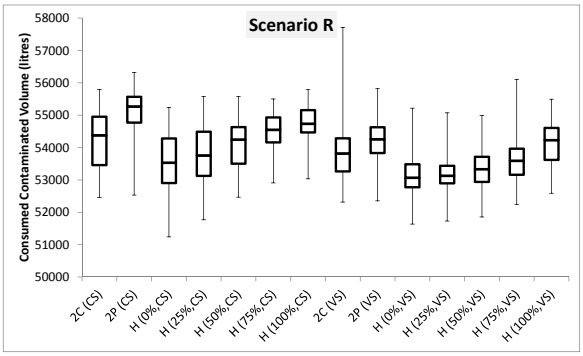

Scenario K
Consumed Contaminated Volume (litres)
45000
44000
43000
42000
41000
40000
39000
38000
37000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

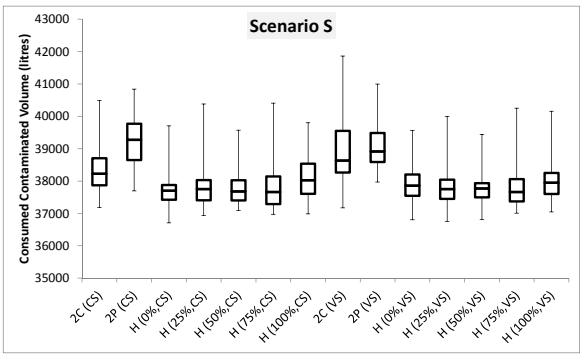

Scenario O
Consumed Contaminated Volume (litres)
50000
48000
46000
44000
42000
40000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

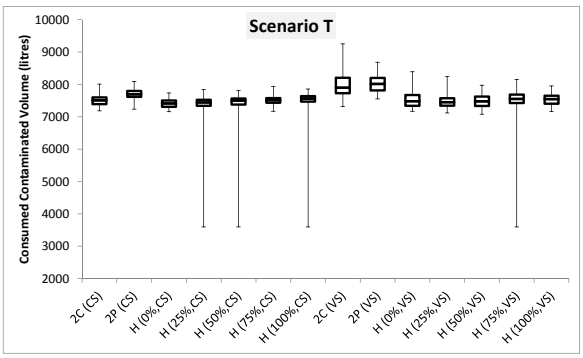

Scenario L
Consumed Contaminated Volume (litres)
49000
48000
47000
46000
45000
44000
43000
42000
41000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario P
Consumed Contaminated Volume (litres)
84000
83000
82000
81000
80000
79000
78000
77000
76000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario Q
Consumed Contaminated Volume (litres)
12000
11000
10000
9000
8000
7000
6000
5000
4000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario R
Consumed Contaminated Volume (litres)
58000
57000
56000
55000
54000
53000
52000
51000
50000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario S
Consumed Contaminated Volume (litres)
43000
42000
41000
40000
39000
38000
37000
36000
35000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario T
Consumed Contaminated Volume (litres)
10000
9000
8000
7000
6000
5000
4000
3000
2000
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

9. 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 14 variants of the various algorithms on 20 scenarios for 500 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 more than 3 years on a single machine. Our approach is able to deal with the more challenging variant of the problem, which allows variable speed. Despite the difficulty given by searching in a much wider feasible region, results show that in most scenarios it is worth to face a much more difficult problem as we are able to compute better quality solutions.

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 behavior, we observed that the new hybrid algorithms which uses a time based encoding outperform the permutation based ones. A significance test confirms this result, with a confidence level below 5%. The use of a MILP solver within the genetic algorithm does not penalize running time, which is dominated by the burden of the simulation component. In future work, we plan to extend the problem and consider a wider set of devices.

References

[1] Addinsoft SARL. XLSTAT v2012.6, 2012. http://www.xlstat.com.

[2] 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.

[3] A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032, 2008.

[4] 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.

[5] J. April, F. Glover, J. P. Kelly, and M. Laguna. Simulation-based optimization: practical introduction to simulation optimization. In *Winter Simulation Conference*, pages 71–78, 2003.

[6] T. Baranowsky and E. LeBoeuf. Consequence management optimization for contaminant detection and isolation. *Journal of Water Resources Planning and Management-Asce*, 132(4):274–282, 2006.

[7] T. Bektas. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[8] A. E. Carter and C. T. Ragsdale. Scheduling preprinted newspaper advertising inserts using genetic algorithms. *Omega*, 30(6):415 – 421, 2002.

[9] 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.

[10] S.-H. Chen and M.-C. Chen. Operators of the two-part encoding genetic algorithm in solving the multiple traveling salesmen problem. In *Technologies and Applications of Artificial Intelligence*, pages 331–336, 2011.

[11] R. Cheng and M. Gen. Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering*, 33(34):761 – 764, 1997.

[12] J. Forrest and R. Lougee-Heimer. *CBC User Guide*. Computational Infrastructure for Operations Research, http://www.coin-or.org/Cbc/cbcuserguide.html.

[13] J. Fowler, S. Horng, and J. Cochran. A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 10(3):232 – 243, 2003.

[14] M. Garey and D. Johnson. *Computers and Intractability*. W.H. Freeman and company, New York, 1979.

[15] 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.

[16] 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.

[17] D. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[18] J. Gonalves, J. Mendes, and M. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167:77–95, 2005.

[19] 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.

[20] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2012. http://www.gurobi.com.

[21] M. Hollander and D. Wolfe. *Nonparametric statistical methods*. Wiley series in probability and statistics: Texts and references section. Wiley, 1999.

[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] C. J. Malmborg. A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research*, 93(1):121 – 134, 1996.

[24] 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.

[25] L. Rossman. *EPANET 2 users manual*. National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, USA.

[26] P. A. Rubin and G. L. Ragatz. Scheduling in a sequence dependent setup environment with genetic search. *Computers & Operations Research*, 22(1):85 – 99, 1995.

[27] F. Sivrikaya-Serifoglu and G. Ulusoy. Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26(8):773 – 787, 1999.

[28] L. Tang, J. Liu, A. Rong, and Z. Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron steel complex. *European Journal of Operational Research*, 124(2):267 – 282, 2000.

[29] P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.

[30] T. Zhang, G. W.A., and M. Smith. Team scheduling by genetic search. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, pages 839–844 vol.2, 1999.