UNIVERSITÀ DEGLI STUDI DI FERRARA

FACOLTÀ DI INGEGNERIA
DOTTORATO DI RICERCA IN SCIENZE DELL'INGEGNERIA
Ciclo XXVII

COORDINATORE Prof. Stefano Trillo
SETTORE SCIENTIFICO DISCIPLINARE ING-INF/05

Solving Real-Life Hydroinformatics Problems with Operations Research and Artificial Intelligence

Dottorando

Dott. Andrea Peano

Tutore

Prof. Marco Gavanelli

Correlatore

Dott.ssa Maddalena Nonato

Anni 2012/2014

Abstract

Many real life problems in the hydraulic engineering literature can be modelled as constrained optimisation problems. Often, they are addressed in the literature through genetic algorithms, although other techniques have been proposed. In this thesis, we address two of these real life problems through a variety of techniques taken from the Artificial Intelligence and Operations Research areas, such as mixed-integer linear programming, logic programming, genetic algorithms and path relinking, together with hybridization amongst these technologies and with hydraulic simulators. For the first time, an Answer Set Programming formulation of hydroinformatics problems is proposed.

The two real life problems addressed hereby are the optimisation of the response in case of contamination events, and the optimisation of the positioning of the isolation valves.

The constraints of the former describe the feasible region of the Multiple Travelling Salesman Problem, while the objective function is computed by a hydraulic simulator. A simulation–optimisation approach based on Genetic Algorithms, mathematical programming, and Path Relinking, and a thorough experimental analysis are discussed hereby.

The constraints of the latter problem describe a graph partitioning enriched with a maximum flow, and it is a new variant of the common graph partitioning. A new mathematical model plus a new formalization in logic programming are discussed in this work. In particular, the technologies adopted are mixed-integer linear programming and Answer Set Programming.

Addressing these two real applications in hydraulic engineering as constrained optimisation problems has allowed for i) computing applicable solutions to the real case, ii) computing better solutions than the ones proposed in the hydraulic literature, iii) exploiting graph theory for modellization and solving purposes, iv) solving the problems by well suited technologies in Operations Research and Artificial Intelligence, and v) designing new integrated and hybrid architectures for a more effective solving.

Sinossi

Molti problemi dell'ingegneria idraulica possono essere modellati come problemi di ottimizzazione vincolata. Spesso nella letteratura dell'ingegneria idraulica tali problemi vengono affrontati tramite algoritmi genetici, sebbene esistano altre tecniche. In questa tesi vengono affrontati due problemi reali dell'ingegneria idraulica attraverso diverse tecniche della Ricerca Operativa e dell'Intelligenza Artificiale, come ad esempio la programmazione matematica, la programmazione logica, gli algoritmi genetici e il path relinking; vengono discusse inoltre delle ibridizzazioni di tali techniche, alcune delle quali con i simulatori simulatori idraulici. Inoltre, questa tesi propone per la prima volta delle codifiche in Answer Set Programming per dei problemi di idroinformatica.

Le due applicazioni reali sono: l'ottimizzazione del piano di intervento in caso di contaminazione della rete idrica, l'ottimizzazione del posizionamento delle valvole di isolamento.

I vincoli del primo problema di ottimizzazione descrivono la regione ammissibile del noto Multiple Travelling Salesman Problem, mentre la funzione obiettivo può essere calcolata da un simulatore idraulico. Questo lavoro propone un approccio di simulazione–ottimizzazione che integra Algoritmi Genetici, programmazione matematica, Path Relinking, e ne discute inoltre un'estesa analisi sperimentale.

I vincoli del secondo problema descrivono un Graph Partitioning arricchito da un problema di flusso massimo, ed è una versione inedita del classico problema di partizionamento. Questo lavoro propone diversi nuovi modelli matematici e logici, sviluppati in Mixed Integer Linear Programming e Answer Set Programming.

Le due applicazioni reali in ingegneria idraulica in esame sono state quindi modellate come problemi di ottimizzazione vincolata permettendo di i) calcolare soluzioni applicabili al caso reale, ii) calcolare soluzioni migliori rispetto a quelle presenti nella letteratura idraulica, iii) sfruttare concetti di teoria dei grafi per la modellazione e la risoluzione dei problemi, iv) risolvere i modelli attraverso l'utilizzo di tecnologie molto efficaci della Ricerca Operativa e dell'Intelligenza Artificiale, v) progettare nuove architetture integrate o ibride che permettono una risoluzione più efficace del problema.

Acknowledgements

I am really grateful to Prof. Marco Gavanelli and Dr. Maddalena Nonato; they have always fostered my path, and I always felt lucky for their constant availability.

I am also grateful to Prof. Marco Franchini and Dr. Stefano Alvisi for their fruitful cooperation and suggestions regarding the hydraulic engineering field.

I am glad to have spent 6 months at the Institute of Computer Science of the University of Potsdam, and I would like to thank Prof. Torsten Schaub, Max Ostrowski, Dr. Martin Gebser, and Prof. Mutsunori Banbara for hosting me and mentoring me during that period.

Thanks to my brothers life and travel companions (increasing order by height): Stefania, Sergio, John, Paola, Stefan. They filled every piece of time I could give them in the past three years.

Thanks to my family, for being the most beautiful and encouraging example of imperfection that works perfectly.

And thanks to Sarah, for every time she has touched my mind.

Contents

List of Figures

List of Tables

List of Acronyms

ACO	Ant Colony Optimisation
AI	Artificial Intelligence
ASP	Answer Set Programming
B&B	Branch and Bound
BIVLP	Bottleneck Isolation Valves Location Problem
CP	Constraint Programming
CLP	Constraint Logic Programming
CLP(FD)	Constraint Logic Programming on Finite Domain
COP	Constrained Optimisation Problem
CSP	Constraint Satisfation Problem
CWS	Contamination Warning System
DFS	Depth First Search
GA	Genetic Algorithm
GPP	Graph Partitioning Problem
FD	Finite Domain
FRP	Feasibility Restoring Problem
IC	Integrity Constraint
IVLP	Isolation Valves Location Problem

LP	Logic Programming
LS	Local Search
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MCB	Minimum Cycle Basis
MFP	Maximum Flow Problem
mTSP	Multiple Travelling Salesman Problem
OR	Operations Research
PMSP	Parallel Machine Scheduling Problem
PR	Path Relinking
SAT	Satisfiability testing
SLD	Selective Linear Definite clause resolution
SB	Symmetry Breaking
RCP	Response to Contamination Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
WDS	Water Distribution System

List of Symbols in Chapter 2

m	number of hydraulic devices (hydrants and valves)	14
n	number of teams of technicians	14
$t^{\mathcal{F}}$	feasible vector of activation times	14
d	depot (or departure point of the teams)	14
τ_{ij}	travelling time from device i to device j	14
U	maximum allowable pause	14
$2C$	*two chromosomes* encoding	22
C_{dev}	chromosome of devices	22
C_{team}	chromosome of teams	22
C_{pause}	chromosome of pauses	24
CS	Constant Speed	25
VS	Variable Speed	25
$2C^{CS}$	$2C$ with constant speed	25
$2C^{VS}$	$2C$ with variable speed	25
C_{part}	second part chromosome of the $2P$ encoding	25
$2P$	*two–part chromosome* encoding	25
$2P^{CS}$	$2P$ with constant speed	25
$2P^{VS}$	$2P$ with variable speed	26
BX	binary crossover	27
$BXPF$	binary crossover with a posteriori feasibility restoring	28
$MILPX$	MILP crossover	28
t	a vector of activation times	29
τ	the matrix of activation times	29
x_{ij}	binary variable stating that a team activates j right after i	29
δ_i	displace variable defined as $t_i^{\mathcal{F}} - t_i$	29
δ	vector of displace variables	29
M	a large number	30
δ_i^+	positive displace variable defined as $\lvert\delta_i\rvert$	30
p_i	the i-th activation time of the parent p	33
$\delta_{i,p}^+$	variable representing $\lvert t_i - p_i\rvert$ if positive	33

$\delta^-_{i,p}$	variable representing $\|t_i - p_i\|$ if negative	33
H	time-based Hybrid GA	34
H^{CS}	H with constant speed	34
H^{VS}	H with variable speed	34
$H(p,s)$	H with $p \in \{0, 25, 50, 75, 100\}\%$ and $s \in \{CS, VS\}$	37
rs	the reference set	51
r	a reference solution	51
g	the target solution	51
$PRseq$	Path Relinking based on sequences	56
PRh	Path Relinking based on activation time (hybrid PR)	57

List of Symbols in Chapter 3

s	a sector, i.e., a set of pipes (a connected component)	65
$v_{a,b}$	the valve that is installed on the extreme a of the pipe (a,b)	66
$v_{b,a}$	the valve that is installed on the extreme b of the pipe (a,b)	66
$ID(s)$	internal demand of the sector s	66
$UD(s)$	unsatisfied demand due to the isolation of the sector s	67
δ_{ij}	user demand of the pipe (i,j)	70
σ	the super–reservoir	70
G	the plain graph representing the hydraulic network	70
V	the set of vertices in G	70
J	the set of junctions nodes in G	70
T	the set of edges connecting σ to the nodes in J	70
E	the set of edges in G	70
W	the set edges' weights in G	70
$\overline{G}$	the extended graph modelling the multicut	72
$\overline{V}$	the set of vertices in $\overline{G}$	72
ϵ_{ij}	a vertex in $\overline{G}$ modelling the generic pipe (i,j)	72
Γ	the set of vertices in $\overline{G}$ containing the ϵ–nodes	72
$\overline{E}$	the set of edges in $\overline{G}$	72
$\overline{\Psi}$	the set of edges modelling the possible places of the valves	72
$\overline{W}$	the set edges' weights in $\overline{G}$	72
S	the set of sectors	72
N_v	the number of available valves	72
N_s	number of allowed sectors	72
τ_{ij}^s	the variable stating if $v_{i,j}$ is a boundary valve of s	73
z_i^s	the partition variable stating if the node i is in the sector s	73
τ^s	the set of binary variables defining the boundary valves of s	76
τ_{ij}^s	the whole set of variables defining the valve positioning	76
$ID(\tau^s)$	satisfied demand of the sector s	76
$UD(\tau^s)$	unsatisfied due to the closure of the valves of s	76
$SD(\tau^s)$	satisfied demand when the valves of s are closed	76

$GP(\tau)$	the partitioning constraints	77
Υ	the whole user demand of the network	77
P	the super–sink	79
$\overline{\overline{G}}$	the graph modelling the network flows	79
$\overline{\overline{V}}$	the set of vertices in $\overline{\overline{G}}$	79
$\overline{\overline{E}}$	the set of edges in $\overline{\overline{G}}$	79
$\overline{\overline{\Psi}}$	the set of high capacitated arcs connecting the ϵ-nodes	79
$\overline{\overline{W}}$	the set of capacities	79
$x^s_{u,v}$	the flow variable for sector s and arc $(u,v) \in \overline{\overline{E}}$	79
$FP(\tau^s)$	the set of flow constraints	83
$UB_{\sharp S}(N_v)$	an upper bound on the number of sector given N_v valves	103
N_v^{min}	minimum number of valves to disconnect the entire network	103
$S_{max}(N_v)$	the maximum number of sectors given N_v valves	104
$Max(\sharp S)$	the optimisation program to compute $S_{max}(N_v)$	104
$\mathcal{C}$	the cycle basis	108

Chapter 1

Introduction

Water is life and since the ancient Sumerians, about 5500 years ago, human beings prefer that water comes to them rather than the opposite. Gorgeous infrastructures had been built later to serve this purpose: Romans constructed about 800 kilometres of water conduits only in the city of Rome. At that time the gravity was the engine to move water, so aqueducts were mainly built to carry this good from the natural sources to the cities and to feed the main places of the city; however connecting common houses was far from possible. Nowadays electricity feeds pumps to boost water into canals and pipes, from the reservoir to our taps. Current infrastructures in developed countries are still enormous and very complex.

Water Distribution Systems (WDSs) can be divided into rural and urban systems. The former distribute water to the primary sector, so also to farms. The latter distribute drinking water to the inhabitants of urban regions. Thus, they are central for human life and their importance is growing hand to hand with the demographic and the economic growth: water is now a rare essential good.

From the design to the management, every phase of the life cycle of a WDS requires huge financial resources, which are rare as well even more in developing countries. At the same time, the trend is to connect any point that demands water, giving rise to capillary infrastructures also called *hydraulic networks*. The task of designing these networks and manage them is now related to sustainability, efficiency, and reliability. A hydraulic network has to be sustainable for the community it serves; it has to be efficient, to avoid wasting useful financial resource; it has to be reliable, because naturally exposed to failures.

Hydraulic engineers take strategic choices during the design phase of urban networks; these choices have immediate implementation costs but they also affect the system's functionality, which means system's costs, until the successive renewing phase. Up to the last decades the existing urban networks grew slowly; so hydraulic engineers could exploit experience and rule of thumb in order to arrange good solutions. Nowadays populations move fast to the cities and local administrations response by setting up big urban districts from scratch. Dubay is an extreme example. Furthermore, hydraulic networks of common cities are getting aged and inadequate by the time; renewing phases of these networks, or their portions, are also more frequent. Finally, hydraulic engineers have to prepare infrastructures and procedures to limit the consequences due to accidental or intentional disasters like contaminations; in fact huge industrial districts are often adjacent to cities, like in Fukushima where radioactive water was spilled out of the nuclear plant; the attention to terrorist threats is increased worldwide after the 9/11, and water supply networks are among the most vulnerable infrastructures and potentially hazardous to the health. In any of these scenarios hydraulic engineers can not find even feasible solutions by hand anymore and need to be supported by automated decision tools.

Hydroinformatics is the discipline that provides computational tools to hydraulic engineers, to support design and management choices for hydraulic networks. It is a hybrid discipline, by definition; it joins scientific concepts in hydraulic engineering and Computer Science indeed.

1.1 Optimisation issues in hydraulic networks

A modern urban WDS consists of thousands of pipes, where pressurized water flows from the reservoir and pumping stations to the users. A pipe can host several users, so its *pressure head* (i.e., the quantity of energy of the water due to the pressure) should be sufficient to satisfy its whole demand in any moment of the day. Several devices are installed on the network, some are devoted to the flow control, others for monitoring or safety purposes; among all the following devices are the most prevalent: *pumps* that increase the energy of the flows, *valves* that regulate the flow can pass through, *quality sensors* that check the quantity of certain elements, *hydrants* that provide water for emergency purposes, *metering stations* that measure the water use of a region. Water flows follow the thermodynamic laws that rule within the pipes the pressure variations, the energy lost by the flows, and the conservation of energy, which are mostly non-linear relations.

To ensure the minimal pressure head given the characterization of any demand point of the network, the diameters of the pipe should be sufficiently dimensioned. On the other hand, bigger diameters lead to higher unit cost of the pipe and energy costs to keep it pressurized. During the second half of the past century the hydraulic community took interest into the problem to find the optimal size of the diameters so that the minimal pressure heads are guaranteed and the network costs are minimized. In the last decades, also reliability has been involved into the optimal design of hydraulic networks, that is the capability of the network to overcome failures and it has been defined analytically by means of the *Resilience Index* [1]. The strategic decisions to achieve the optimal design of the network are the diameters of pipes; not every choice is feasible, in fact diameters directly affect water flows in the network, and by consequent of the thermodynamic laws also the pressure, which should respect minimal requirements depending on the user demands. Diameters also affect costs, that should be minimized, and lead to a certain degree of resilience of the network, that should be maximized. Recently the trend in hydroinformatics is to address this problem by metaheuristics; for example Creaco and Franchini exploited Genetic Algorithms (GAs) [2], Cunha and Sousa used Simulated Annealing [3], and Maier et al. used Ant Colony Optimisation (ACO) [4]. A literature review is provided in [5].

Apart from this, also the task to position hydraulic devices on the network is a strategic decision. Pumps are installed to guarantee pressure heads by minimizing the pumping system's costs [6]. Valves are installed to allow for isolation of broken pipes by minimizing both the unsatisfied demand due to the isolation and the costs of the isolation system's costs [7, 8]. Sensors are placed by seeking a compromise between the coverage of the network, in order to detect as many contamination events as possible, and their costs [9] (see [10, 11] for a literature review).

As far as the operative problems, they are more related to schedule the activities of these devices. For example pumps and valves can be controlled in order to increase or decrease pressure during some time periods [12–14] in order to save energy. Valves and hydrants can be activated in certain instants to reduce the impact of contamination events [15, 16].

Many other design and management issues in hydraulic network are optimisation problems, and hydroinformatics provides automated tools to address them and find good solutions. For example, all the above optimisation issues can be solved as combinatorial problems in which a set of *decision variables* with discrete domains draws the *solution space* of the problem; dedicated algorithms are used to search optimal or sub-optimal solutions on that space. For example, in the optimal design of the diameters, every pipe becomes a discrete variable and its domain is the set of possible diameters; every possible position of the pipe can be represented by a boolean variable, stating whether the device is installed or not; the schedules of the devices can be represented by discrete variables containing the actual activation times. Combinatorial problems can be addressed straightforwardly by means of evolutionary metaheuristics, and this is also a trend within the hydraulic community, that widely uses GAs among all; the studies cited so far are GA-based indeed. This trend is motivated by many factors. GAs are well integrated into common software suits like MATLAB [17]; also, these techniques allow for multi-objective optimisation, which is an important requirement if the aim is to strike a good compromise between functionality and costs. Moreover, GAs are very well suited to be coupled with hydraulic *simulators*, that are often used to check the feasibility of the solution and eventually to compute its quality. Due to the nature of the thermodynamic laws in fact, it is not straightforward to formalize the relations between decisions of the problem and the consequent effects on the water flows. Hydraulic simulators, like EPANET [18], allow for an over-time estimation of the water flows in terms of pressure and speed. In particular EPANET takes in input a description of the network topology (like position and length of pipes), the user demands in any demand point and for several time periods, in order to handle with dynamic variations of user demands along the day; it computes instant by instant the state of the network in any point. In EPANET the values of the decision variables can be easily integrated and become an additional input to the simulation: estimating the effect of those decisions is only a matter of computational time. The resulting architecture configures a classical *simulation–optimisation* approach [19, 20], where metaheuristic procedures use the simulator as a black-box function.

In combinatorial optimisation a solution of the problem is represented by one or more combinations of values the variable can take, called *strings*. The the whole set of possible combinations is the *search space* that an algorithm can explore to find out the best solution, and its cardinality measures the size of the search space. For instance, recall that in the optimal design problem every pipe is a discrete variable spanning over the possible diameter values, so having m pipes and k possible diameters the strings that can be represented are k^m: an exponential number. Metaheuristics like GAs have the property to explore only a part of this search space, turning out with good solutions if well calibrated. In general, a search strategy should be tuned to have a good compromise between how much search space it explores and the quality of the best solution it is able to compute.

However, depending on the problem's structures the researcher is faced with, the decision variables are often linked together by some relations that make many solutions infeasible. In the literature these relations are called *constraints*, and their role is fundamental to understand the exact nature of the problem and its actual complexity. Metaheuristics are blind w.r.t. these constraints and can even get stuck into infeasible regions of the search space. Sometimes hydraulic engineers can not even find a way to express and exploit these constraints and they often delegate to *a posteriori* procedures the task to compute whether a solution is feasible or not; this assessment is made possible by a burdensome hydraulic simulation in the case a dedicated algorithm is not available.

1.2 Constrained Optimisation Problems: a practical approach

Computer Science provides further modelling and solving tools that can be very effective to address the very same optimisation problems in hydraulic network. In the literature of Operations Research and Artificial Intelligence, optimisation problems that exhibit constraints on variables are called Constrained Optimisation Problem (COP). Optimisation problems can be often represented by:

- a set v of variables with a finite domain of real or integer values;
- a set c of constraints linking finite subsets of variables;
- an objective function $o(w, v)$ that associates costs (or weights) w to the variables and has to be minimized or maximized.

A variable is said to be assigned when it takes one value of its domain. An assignment $\overline{v}$ for the entire set v that respects the whole set of constraints is a *feasible solution* of the problem. A solution $\overline{v}'$ is *optimal* for minimization (or maximization) problems if $o(w, \overline{v}') \leq o(w, \overline{v}'')$ (or $o(w, \overline{v}') \geq o(w, \overline{v}'')$) for any other solution $\overline{v}''$.

Basically a constrained optimisation problem maps the decisions of the problem into the variables, the intrinsic relations among the decisions into constraints, and the goodness of the decisions into an objective value computed by a cost function.

Moreover, Computer Science provides deep theoretical knowledge to study every aspects of optimisation problems. For example, complexity theory studies how difficult is the task to solve computational problems and in particular COPs [21]. Graph theory studies those problem structures that can be represented into a graph [22, 23], that is a natural way to model networks in mathematics. Two main fields actively put effort into finding new effective theories, languages, and algorithms to address COPs and their applications: Operations Research (OR) and Artificial Intelligence (AI).

In Operations Research the best known paradigm to declare constrained optimisation problems is the *mathematical programming*; the constraints are linear and non linear equations, linking continuous and integer variables. For problems having discrete and real variables and linear constraints this paradigm is called Mixed Integer Linear Programming (MILP), whereas for non-linear constraints is Mixed Integer Non-Linear Programming (MINLP). The resolution of a MILP program relies on the mathematical theory of the polytopes [24], a geometric representation of the problem. To compute the optimal solutions of these programs the best known solvers apply, among all, Branch and Bound (B&B) and the *Simplex* algorithm. Also network problems on graphs can be effectively expressed with mathematical formulations. Moreover, metaheuristics belong to this field either.

Several communities in Artificial Intelligence are active in developing technologies to solve optimisation problems. The principal formalisms used in AI are Constraint Programming (CP) [25] and Logic Programming (LP) [26]. The former is a declarative paradigm that divides constraints into families, e.g., scheduling constraints, geometrical constraints, and each particular constraint has an ad hoc *propagation* algorithm. The latter is a declarative paradigm based on the *first-order logic*, that expresses relations among variables by means of logic rules. Well known specialization of LP very suited to address optimisation problems are Constraint Logic Programming (CLP) [27] and Answer Set Programming (ASP) [28]. The core resolution strategies here rely basically on the concept of propagation, which means that a choice on a variable, i.e., deleting one or more values from its domain, causes a modification on the variables' domains that are linked by the same constraint. So that choice, thanks to a dedicated algorithm, is propagated towards other variables. The aim is to keep consistent the domains after any choice. The roots of these techniques were developed to solve the well known Constraint Satisfation Problem (CSP) [29] and Satisfiability testing (SAT) [30].

Several approaches in OR have been already proposed to solve optimisation problems in hydraulic network. For example recently MINLP has been proposed for the optimal design of diameters [31]. Several MILP programs were proposed for the optimal placement of pumps [6] and sensors [32, 33]. Anyway, mathematical programming is not new for the hydroinformatic community. More recently solution approaches have been proposed in AI either, for example for the optimal placement of valves [34].

Knowing the actual structure of the problem through its constraints means to be able to design better solving architectures, either exact or not. The constraints in fact interpret the features the solution should have, thus allow for computing only applicable solutions. Also, constraints give a certain shape to the problem, they determine the family it belongs, thus far giving an appropriate name to the problem, which makes possible to identify the pertaining literature in Computer Science. If complete formal description does exist already, the constraints may allow the researcher to identify certain *substructures* of the problem; for example it may be possible to divide the constraints into different subsets, each one having the form a particular problem family. Moreover, constraints and variables might be *decomposed* into hard to solve and easy to solve substructures, that makes possible an integrated solving procedure. Finally, the language and the solver to be considered to solve a problem may depend strongly on nature of the constraints. For example, in case of integer variables and *global* constraints, i.e., constraints spanning over large sets of variables, one could choose for a CP approach; in case of continuous variables and linear equations one could choose for a MILP approach; in case of logic implications, one could choose for ASP.

In other words, modelling design and management issues in hydraulic networks by means of constrained optimisation problems and addressing them by the appropriate techniques can have many benefits: detect the actual structure of the problem and its complexity and access to the available literature and solving technologies; consider only feasible solutions and drive the search by means of an analytical description of the objective function, which allows for saving the computational cost of burdensome simulations; enrich existing metaheuristic approaches with ad hoc procedures depending on the constraints; avoid to exploit the combinatorial component of a problem that is easier to solve by a systematic approach. We believe that many problems in hydroinformatics have not been formulated as constrained optimisation problems yet, even though they had been investigated by many studies.

The applications this work describes are real–life problems in hydroinformatics. The first is operative and optimises the tasks of the technicians in case of contamination events of the urban hydraulic network. It is called Response to Contamination Problem. The second is strategical and optimises the positioning of the isolation valves to make broken pipes isolable. It is called Isolation Valves Location Problem. Next sections introduce these problems and give a general overview about how a proper identification of the constrained substructures enhanced the state of the art.

1.2.1 Handling contaminations

During contamination events of the WDS users could consume toxic water until the contaminant is completely worn out. Depending on the toxicity of the agent, contaminated water can be a health hazard even mortal. Quality sensors monitor the quantity of certain chemical and biological elements and raise an alarm whenever an element overcomes a safety threshold. The sensor also gives an information about where the contamination is active. From this, the technicians can apply a response by activating the valves and the hydrants installed on the surroundings. Opening hydrants means dispelling contaminated water and decreasing the pressure heads of the pipes. Closing valves means diverting contaminated water, either towards less urbanised districts (with less user demand) or towards open hydrants, for example.

The problem to optimise the placement of the sensors and to identify the set of devices to be activated in case of contamination events was addressed in [15]. Here the response was computed by considering only a restricted set of "reasonable" schedules to be assigned to the available technicians. The study was tested into the network of Ferrara, a medium sized city of about $100,000$ citizens. The quality of the response was measured with the volume of contaminated water consumed by the population, and it was estimated by a hydraulic simulation that takes about 5 seconds. A genetic algorithm was used to optimise the set of devices to be activated.

Actually the technicians can be much less than the number of devices and they would have to travel from device to device to operate the designated operations. Knowing this, the schedules computed so far are not applicable anymore on a real context. Fortunately, the feasible region of this problem is described by a well known constrained optimisation problem, namely the Multiple Travelling Salesman Problem (mTSP), which is NP-hard. In the mTSP, m salesmen visit once and only once every customer they have to serve. In this application the salesmen are the technicians and the cities are the hydraulic devices. The constraints of the Travelling Salesman Problem (TSP) describe all those schedules that allow the technicians to reach their devices compatibly with the travelling times on the street network. Unfortunately the interactions between the activation times and the water flows are too complex to formalize an appropriate objective function, so any solution of the mTSP should be evaluated through a hydraulic simulation. A first attempt would be then to apply a common sense criterion in case of emergency: instruct the technicians in order to activate the devices as soon as possible; this can be translated into a mathematical relation that minimizes, for example, the *makespan*, i.e., the activation time of the last device. The hydraulic simulation of this common sense schedules gives volumes of contaminated water consumed by the citizens much greater than the schedules computed randomly.

This has incentivized to design search algorithm based on evolutionary templates that could exploit the knowledge of the particular feasible region of the mTSP. The resulting architectures exploit effectively the combinatorial nature of the problem and its constraints, as described in Chapter 2; they combine genetic encodings for the mTSP, MILP models based on the mTSP, and EPANET. The mTSP substructure of the Response to Contamination Problem has also made possible to develop intensification procedures based on Path Relinking.

1.2.2 Placing isolation valves

The positioning of isolation valves is a strategic decision, very related to the resilience of the network. Every time a pipe gets broken, technicians close a group of valves to isolate that zone and fix the damage. Ideally every pipe should have two valves at its extremes; unfortunately, a part the installation costs, the valves is also a breakable object and lead to higher probability of pipe failures and consequent maintenance costs. The isolation system should be designed limiting the number of available valves, typically much lower than the number of pipes. Given that, the valves should be installed so that any pipe can be isolated; this determines that a feasible positioning of valves draws a *sectorization* of the network. Every *sector* is a group of pipes that get isolated all together every time one of them needs to be fixed. The users linked to that sector experience the lack of water; this service disruption can be measured by considering the instant water demands (l/s) of all the users that get disconnected during the isolation. So the optimal positioning of the available isolation valves would minimize a function on the service disruption and the valves' costs; this problem is called Isolation Valves Location Problem (IVLP). A particular specialization minimizes the unsatisfied demand in the worst isolation case, and it is called Bottleneck Isolation Valves Location Problem (BIVLP).

The combinatorial component of this problem can be drawn by considering a boolean variable for each place that can host a valve. Typically valves can be installed on the two extremes of the pipes, so with m pipes and n valves the combinations are $\binom{2m}{n}$. Actually not every combination is feasible, because many placements of the valves do not draw a sectorization of the network, which means that some pipes cannot be isolated. Moreover, some placement of the n valves are "inefficient" in the sense that the sectorizations are given by a subset of those valves, and the others are useless. The genetic algorithms in the literature explore these solutions [7]; fortunately given a positioning of valves it is quite simple to detect the sectors and the unsatisfied demand they determine, so dedicated postprocedures detect infeasible solutions and computes the fitness of the feasible ones.

The first exact approach for this problem was a Constraint Logic Programming on Finite Domain (CLP(FD)) [34], so in AI. This program defines the constraints that allow for the actual sectorization of the network and prevent to install useless valves; the resolution was the first providing the exact Pareto front for the *Apulian* network used in [7].

Chapter 3 describes thoroughly the first mathematical formalization of the IVLP. This formalization implements a mathematical description of the well known Graph Partitioning Problem, which is NP-hard. Also, to compute the unsatisfied demand of the various isolation scenarios a Maximum Flow Problem structure has been integrated into the model. This model describes mathematically the two main structures of the problem: the partitioning, and the flows. The first layer involves the *hard* variables of the problem, because they are the decision variables as well as integer; the nested layer involves the *easy* variables, concerning the water flows. This perspective allows for the resolution by decomposition methods. This work also investigates another approach in Logic Programming, more precisely in Answer Set Programming. In ASP the problem can be modelled in several ways, for example the sectorization can be achieved either by explicitly defining sectors or not; also, the unintended isolations can be modelled by computing the reachability of pipes as well as of sectors from the tank.

Chapter 2

Scheduling countermeasures in case of contaminations

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 farms. 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 a 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 neighbourhoods 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 [35]. 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.

2.1 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 2.2). 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. Three issues characterize this problem.

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. A hydraulic and quality simulator such as EPANET [18] takes as input the network configuration, the user demands along the day, 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. Several toxic substances can cause human death even when present at low concentration, e.g., $0.3mg/ml$. Therefore, the period to be considered as the simulation horizon 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 user 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, ranging from $5''$ to $7''$ 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 [15]. 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 has already being flowing through the network for a while. Therefore, the objective is not to operate all devices as soon as possible, 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, can 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

$$t_i^{\mathcal{F}} + \tau_{ij} = t_j^{\mathcal{F}} \tag{2.1}$$

where τ_{ij} is the travelling time required to cover the distance $dist_{ij}$ from the location of device i to j, with respect to a given constant speed. All schedules complying with these requirements are feasible, and the search for an optimal schedule must take such constraints into account. Note that the feasibility constraints of the scheduling coincide with those of a well known combinatorial optimization problem, i.e., the Multiple Travelling Salesman Problem (mTSP) [36].

As stated before, devices activation strongly affects how water flows on a pipe network -and thus how contaminant spreads - by increasing or decreasing the discharge within some pipes and towards certain nodes (e.g. opening a hydrant typically increases the flow towards nodes next to the hydrant itself); delaying a valve or an hydrant opening can thus avoid the contaminant to be drawn toward a critical zone (e.g. a densely populated zone). Thus, delaying some devices' activations might yield more effective schedules. Such delays can be modelled through a "maximum allowable pause" once technicians have arrived on site before each device activation; in this case, $t_j^{\mathcal{F}}$ can not exceed $t_i^{\mathcal{F}} + \tau_{ij}$ more than a given quantity U, so Eq. (2.1) becomes:

$$t_j^{\mathcal{F}} \geq t_i^{\mathcal{F}} + \tau_{ij} \tag{2.1a}$$

$$t_j^{\mathcal{F}} \leq t_i^{\mathcal{F}} + \tau_{ij} + U \tag{2.1b}$$

Allowing for a pause before each device activation has then three important returns w.r.t. our problem:

- inter-relations among the activation of different devices can be captured;
- the *variable speed* of technicians is also modeled, because the quantity $dist_{ij}/(t_j^{\mathcal{F}} - t_i^{\mathcal{F}})$ is not always the same;

- the feasible region is enlarged since the number of feasible activation times increases remarkably.

It is worth noting that the scheduling horizon (the range of activation times) here considered depends on the selected devices and the associated distances. In particular, for the constant speed case, the scheduling horizon is bounded by the duration of the longest Hamiltonian path and it is much shorter than the simulation horizon. In the variable speed case, in principle, the scheduling horizon could be as long as the simulation horizon, but in practice there is no point in operating a device long after the alarm was raised. In fact, since contaminant concentration rapidly decreases over time (though still significant along the simulation horizon), decisions concerning devices activations taken during the initial period have more impact. As stated above, this does not mean that all devices should be activated as soon as possible, however postponing their activation for too long leaves high concentrations of contaminant all over the network for too long causing high level consumption of contaminated water. Therefore, as far as the the variable speed assumption, the scheduling horizon becomes considerably longer than the constant speed case, though still much shorter than the simulation horizon.

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 not 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.

Glover in [19] 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 the effect of small modifications with respect to a given schedule, such as postponing or anticipating the operation time of a single device, can not be evaluated a priori. Therefore, there is no computational saving in searching within a neighbourhood 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 neighbourhood based local searches, for their ability to explore a wider part of the feasible region with a limited number of solution evaluations. To computationally support this intuition a standard Local Search algorithm has been implemented and tested. Such considerations lead us to select Genetic Algorithm (GA) as the solution approach.

The solving approach consists of 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). Section 2.2 discusses related works.

The general framework architecture of the genetic algorithms is discussed in Section 2.3. The first two encodings come from the literature on the genetic algorithms for the mTSP, namely the *two chromosome* representation described in Section 2.4 and the *two part chromosome* representation presented in Section 2.5. The third, i.e., the *time-based* representation introduced in Section 2.6, is original and it is built on the mathematical models, in Mixed Integer Linear Programming (MILP), developed for the mTSP. We experimentally compare all these representations on the real instance of a medium sized city. Results are presented and discussed in Section 2.7.

The first time-based representation was presented in [37]. Afterwards, a further study [38] extended the previous one by:

- extending every genetic encoding to handle variable speed of the technicians;
- dimensioning the population size through dedicated experiments;
- evaluating pros and cons of using high performance MILP solvers in the framework of the time-based representation;
- performing a comprehensive computational campaign spanning all over the proposed solving methods and the contamination scenarios.

A preliminary report of these results was first presented in [39].

This chapter wants to be an overview of the studies above, which are the state of the art for the Response to Contamination Problem (RCP); finally it extends the genetic approach with a final intensification procedure by Path Relinking, which is thoroughly described in Section 2.8.

2.2 Related Works

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

Environmental studies suggest to use several types of quality sensors simultaneously [40]. The hydraulic engineering literature provides many contributions that report the use of optimization techniques, up to various degrees, to tackle two problems strongly related to the 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 the RCP. Concerning sensor location when designing a CWS, Murray et al. [35] propose a MILP model which is close to the p-median facility location problem, where sensors are associated to facilities, the contamination accidents 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 instantaneously, 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. Recently, Krause et al. [41] 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: Alfonso et al. [16] and Guidorzi et al. [15] propose a multi-objective approach minimizing the number n of operated devices and 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. Baranowsky and LeBoeuf [42] employ a genetic algorithm to determine the optimal flushing and valving operations in order to minimize the total network contaminant concentration following sensor detection. However, as well as Alfonso et al. [16], they suppose to activate all the selected devices instantaneously and simultaneously. More realistically, Guidorzi et al. [15] build 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.

Recently, Shafiee and Berglund [43] proposed a simulation-optimisation approach to compute optimal sensor-hydrant decision trees to be browsed during the contamination events. However the travelling times of technicians are not taken into account, then the application of the resulting hydrant schedule may be impracticable.

The feasibility constraints of the RCP coincide with those of a well known combinatorial optimization problem, i.e., the multiple Traveling Salesman Problem (see [36] 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 the 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, local changes of the mTSP solution concerning just a few arcs of the routes affect the solution value only for the modified arcs, while in RCP the entire water flow can be modified by a local change, and the impact evaluation of any small change requires an entire hydraulic simulation. In addition, 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 model 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 [44]. 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 [45]. Both representations are heavily affected by redundancy. Carter and Ragsdale [46] propose 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 [47]. More recently, Chen and Chen [48] experimentally analyze different crossover and mutation operators for the Two Part Chromosome on few instances of the TSPLIB [49].

The RCP feasibility structure is common to another well known family of sequencing problems in combinatorial optimization, i.e., the Parallel Machine Scheduling Problem (PMSP) with sequence dependent set up times. 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, genetic representations are affected by a certain degree of redundancy, so that similar solutions may be represented by quite different individuals, 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. For example, Fowler et al. [50] use a genetic algorithm to assign jobs to machines, while machine scheduling on the individual machine is performed according to a greedy criterion. Gonçalves et al. [51] use a random keys representation 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. Sivrikaya-Şerifoğlu and Ulusoy [52] tackle 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 Zhang et al. [53], multiple teams of photographers must be scheduled to serve a large number of schools. 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. Cheng and Gen [54] devised a hybrid genetic algorithm 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 [55] which solves a scheduling problem with set up times minimizing total tardiness.

Many other references can be found in Allahverdi et al. [56], 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.

Finally, several techniques for *simulation–optimisation* problems have been proposed in the literature recently [20,57]. However, these studies address unconstrained problems.

2.3 A Genetic Algorithm Framework for the Response to Contamination Problem

A genetic algorithm (GA) draws inspiration from the mechanism of species evolution; each solution of a problem is encoded in some data structure, the *chromo-*

some. 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 the RCP, the evaluation of an individual's 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 [18], 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 its concentration for each time slot and location, whence the quantity of consumed contaminant 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, at which concentration and in which instants it 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, guaranteeing that at least one device is assigned to each team. For the variable speed case, a third vector of n elements 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.

2.4 A Genetic Algorithm Based on Sequences

As mentioned, the 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 [46], and we will use the acronym $2C$ in the rest of the chapter. 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:

$$\begin{array}{c} \overbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 3 & 4 & 1 & 2 & 8 & 5 & 7 & 6 \\ \hline \end{array}}^{C_{dev}} \\ \underbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 1 & 3 & 2 & 3 & 2 & 2 \\ \hline \end{array}}_{C_{team}} \end{array} \tag{2.2}$$

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 [46]. For example, if we permute columns 1 and 2 in Eq. (2.2), 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 $n!m^n$ [46].

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* [58]. Given two parents, f and m, and an integer i in the interval $[1, n]$, two offspring 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; the second one takes the first i columns from m and the others from f in the given order. In the example depicted in (2.3), 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.

$$
\begin{array}{rcl}
\mathrm{f} & = & \begin{array}{c}\begin{array}{|c|c|c|c|c|c|c|c|}\hline 6&4&1&2&8&5&7&3\\\hline\end{array}\\[4pt]\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&2&1&3&2&1&2&2\\\hline\end{array}\end{array}\\[12pt]
\mathrm{m} & = & \begin{array}{c}\begin{array}{|c|c|c|c|c|c|c|c|}\hline 2&7&3&8&4&6&1&5\\\hline\end{array}\\[4pt]\begin{array}{|c|c|c|c|c|c|c|c|}\hline 2&1&3&3&1&2&1&2\\\hline\end{array}\end{array}\\[12pt]
 & \Rightarrow & \underbrace{\begin{array}{c}\begin{array}{|c|c|c|c|}\hline 6&4&1&2\\\hline\end{array}\\[4pt]\begin{array}{|c|c|c|c|}\hline 1&2&1&3\\\hline\end{array}\end{array}}_{f}\underbrace{\begin{array}{c}\begin{array}{|c|c|c|c|}\hline 7&3&8&5\\\hline\end{array}\\[4pt]\begin{array}{|c|c|c|c|}\hline 1&3&3&2\\\hline\end{array}\end{array}}_{m}
\end{array}
\tag{2.3}
$$

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 2.1 shows the tree representation of the offspring in (2.3): 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.

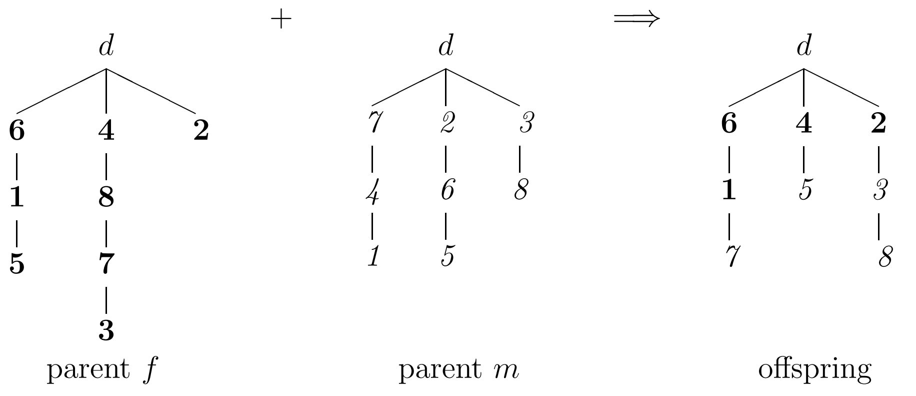


Figure 2.1: An example of the tree representation of the two parents and the offspring in (2.3) obtained by the crossover

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 offspring from very similar parents, we adopt a standard team naming approach: the team operating device 1 takes name 1; the team that operates the device with smallest identifier amongst the remaining devices takes name 2, and so on.

Allowing for Variable Speed in $2C$

As already mentioned, in the 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* (VS). The resulting encoding consists of three chromosomes, but, for the sake of clarity, we call it "*Two Chromosomes encoding with Variable Speed*" ($2C^{VS}$). We will refer to the *Constant Speed* (CS) variant as $2C^{CS}$.

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 (2.2), for an opportune travelling time matrix τ we may obtain that the team 1, moving at constant speed, 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 (2.2) in the space of $2C^{VS}$:

$$\begin{array}{c}
\begin{array}{|c|c|c|c|c|c|c|c|}\hline 3 & 4 & 1 & 2 & 8 & 5 & 7 & 6 \\ \hline \end{array} \\
\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 1 & 3 & 2 & 3 & 2 & 2 \\ \hline \end{array} \\
\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 1 & 5 & 5 & 4 & 3 & 3 & 2 \\ \hline \end{array}}_{C_{pause}}
\end{array} \tag{2.4}$$

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 $t_5^{VS} = \tau_{d2} + 5 + \tau_{25} + 3 = 16$. 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.

2.5 A Genetic Algorithm based on the Two-part Chromosome

In [46] a so called *two-part chromosome* ($2P$, or $2P^{CS}$) 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 (2.2) would become (2.5).

$$\underbrace{\overbrace{\begin{array}{|c|c|}\hline 3 & 1 \\ \hline\end{array}}^{2}\overbrace{\begin{array}{|c|c|c|c|}\hline 4 & 8 & 7 & 6 \\ \hline\end{array}}^{4}\overbrace{\begin{array}{|c|c|}\hline 2 & 5 \\ \hline\end{array}}^{2}}_{C_{dev}} \quad \underbrace{\begin{array}{|c|c|c|}\hline 2 & 4 & 2 \\ \hline\end{array}}_{C_{part}} \tag{2.5}$$

This way, the size of the representation space is lowered to the order of $n!\binom{n-1}{m-1}$. As in [46], 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. Actually, in [46] the first chromosome of the new child is coupled to every second part chromosomes in the population in order to find out a better solution; this search would consume a number of simulations in the amount of the population size for each individual of the new population. This procedure should be avoided in this application, as there is a limit on the EPANET calls.

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. (2.5) could also be represented in $2P$ as follows:

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

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 (2.4) would become the following:

$$\begin{array}{ll} \overbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 3 & 1 & 4 & 8 & 7 & 6 & 2 & 5 \\ \hline\end{array}}^{C_{dev}} & \overbrace{\begin{array}{|c|c|c|}\hline 2 & 4 & 2 \\ \hline\end{array}}^{C_{part}} \\ \underbrace{\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 5 & 1 & 4 & 3 & 2 & 5 & 2 \\ \hline\end{array}}_{C_{pause}} & \end{array} \tag{2.6}$$

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 the $2C^{CS}$ (i.e., the basic $2C$), $2C^{VS}$, $2P^{CS}$ and $2P^{VS}$ GA s, 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.

2.6 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 the i^{th} gene modelling the 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

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

$$m = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 8 & 1 & 3 & 6 & 3 & 2 & 2 \\ \hline\end{array}$$

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

$$c = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 3 & 8 & 3 & 3 & 8 & 3 & 7 & 2 \\ \hline\end{array}$$

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 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 rise to a second approach denoted as $MILPX$.

2.6.1 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 travelling time matrix τ:

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

Vectors $m = [1, 1, 4, 8]$ and $f = [2, 5, 1, 1]$ model feasible schedules but the BX 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 [36] and routing problems in general, among which the following *2-index flow-based* formulation [59]. The constraints extend the mTSP model with travelling times information, the objective function minimizes the distance of the vector $t^{\mathcal{F}}$ from t. To make it possible additional variables and constraints have to be integrated into the MILP model; this section describes first the core of

the MILP model for the mTSP, then it extends it with the constraints that are necessary to generate vectors $t^{\mathcal{F}}$ that are as close as possible to the input.
The input parameters are:

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

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 constraints:

$$t_i^{\mathcal{F}} \geq \tau_{di} \qquad \forall i \in \{1..n\} \tag{2.7a}$$
$$\delta_i = t_i - t_i^{\mathcal{F}} \qquad \forall i \in \{1..n\} \tag{2.7b}$$
$$t_d^{\mathcal{F}} = 0 \tag{2.7c}$$
$$\sum_{i\in\{1..n\}} x_{di} = m \tag{2.7d}$$
$$\sum_{j\in\{1..n\}\cup d} x_{ij} = 1 \qquad \forall i \in \{1..n\} \tag{2.7e}$$
$$\sum_{j\in\{1..n\}\cup d} x_{ij} = \sum_{h\in\{1..n\}\cup d} x_{hi} \qquad \forall i \in \{1..n\} \tag{2.7f}$$
$$t_i^{\mathcal{F}} \leq M + x_{di}(\tau_{di} - M) \qquad \forall i \in \{1..n\} \tag{2.7g}$$
$$t_i^{\mathcal{F}} + \tau_{ij} \leq t_j^{\mathcal{F}} + (1-x_{ij})M + x_{ji}(\tau_{ij}+\tau_{ji}-M) \qquad \forall i,j \in \{1..n\} \tag{2.7h}$$

Constraint (2.7a) says that device i can be activated no earlier than the time it takes to reach it from d. Eq. (2.7b) is the definition of δ. Teams leave the depot at time 0 (Eq. (2.7c)). All m teams depart from the depot (Eq. (2.7d)). All nodes except d are visited exactly once (Eq. (2.7e)). For each node i, the total number of teams arriving to i is equal to the number of teams leaving i. Eq. (2.7f) are the so called flow balance constraints. Constraint (2.7g) is the linearisation of the implication $x_{di} = 1 \Longrightarrow t_i^{\mathcal{F}} \leq \tau_{di}$, where M is a sufficiently large positive number; together with Constraint (2.7a), it imposes that the starting time of the first devices must be equal to their traveling time from d. Constraint (2.7h) links the activation times $t^{\mathcal{F}}$ to the ordering between devices given by matrix X; indeed, (2.7h) 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. (2.7h) imposes that the arrival time at device j equals the starting time from i plus the travelling time from i to j, thus implementing the constant speed variant of the time-based GA. The objective function associated to problem (2.7a-2.7h) is the minimization of the L_1 distance between $t^{\mathcal{F}}$ and t, namely:

$$min||\delta||_1 = min\left(\sum_{i\in\{1..n\}} |\delta_i|\right) \tag{2.8}$$

To linearize this function, we introduce new unknowns δ^+ that represent the absolute value of δ, and minimize their sum:

$$min||\delta||_1 = min\left(\sum_{i\in\{1..n\}} \delta_i^+\right) \tag{2.8a}$$

$$\delta_i^+ + \delta_i \geq 0 \qquad \forall i \in \{1..n\} \tag{2.8b}$$

$$\delta_i^+ \geq -\delta_i \qquad \forall i \in \{1..n\} \tag{2.8c}$$

We call this problem the *Feasibility Restoring Problem* (FRP).

Problem Complexity

Theorem 1. *The* Feasibility Restoring Problem, *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. The Feasibility Restoring Problem (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 [21].

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 travelling times on the network $\{\tau_{ij}\}$, and initialization set up times correspond to the travelling 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 linearisation, motivates the choice of the L_1 norm to evaluate the distance between $t^{\mathcal{F}}$ and t.

Summarizing, since FRP is a reformulation of a NP-Hard machine scheduling problem, it follows that FRP is NP-Hard as well. $\square$

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 (2.7a-2.7h) and solving this model by a MILP solver.

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 (2.7g) and (2.7h) are now modified as follows:

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

2.6.2 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 vertices, and with sides parallel to the coordinate axes. Figure 2.2) shows a 3D representation of this space; crosses represent feasible points, m and f are the mating individuals of size 3; c is the closest feasible point (at distance δ) to a vertex of the parallelepiped. 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 (2.7a-2.7h), 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 (2.7b) of the displacement δ becomes:

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

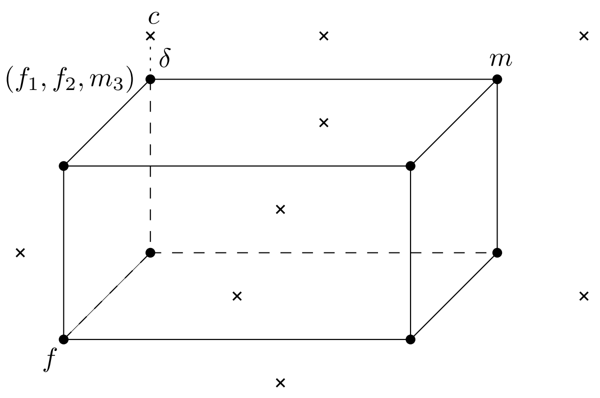


Figure 2.2: Graphic representation of the crossover $MILPX$ in a 3D space

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 and constraints are introduced. Variable $\delta^+_{i,p}$ model the $i-th$ distance between the offspring and the parent p, if positive. Variable $\delta^-_{i,p}$ model the $i-th$ distance between the offspring and the parent p, if negative. Thus, for each parent p (m and f) and device i the following relations should hold:

$$\delta^+_{i,p} = \begin{cases} 0 & \text{if } |t_i - p_i| < 0 \\ |t_i - p_i| & \text{if } |t_i - p_i| \geq 0 \end{cases}$$

$$\delta^-_{i,p} = \begin{cases} |t_i - p_i| & \text{if } |t_i - p_i| < 0 \\ 0 & \text{if } |t_i - p_i| \geq 0 \end{cases}$$

A possible linearisation of the equations above is:

$$\begin{aligned} \delta^+_{i,p} &\leq Mb_{i,p} && \forall i \in \{1..n\} \\ \delta^-_{i,p} &\leq M - Mb_{i,p} && \forall i \in \{1..n\} \\ p_i &= \delta^+_{i,p} - \delta^-_{i,p} + t_i && \forall i \in \{1..n\} \end{aligned}$$

The variables b^p_i are binary and state whether the distance between the parent and the offspring is positive or not. The following constraints require such distances to be greater than a positive threshold Δ:

$$\sum_{i \in \{1..n\}} \left(\delta^+_{i,p} + \delta^-_{i,p}\right) \geq \Delta$$

Note that the same condition can not be guaranteed by simply bounding the number of coordinates coming from each parent, i.e.,

$$\sum_{i \in \{1..n\}} w_i \geq 1$$

and

$$\sum_{i \in \{1..n\}} w_i \leq n - 1$$

For example, these conditions have 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 2.7.

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* (H) with constant and variable speed (H^{CS}, H^{VS}). The initial population is built by using the same procedure as the sequence-based encodings (see Section 2.3): for each new individual a random permutation of devices is associated with a random combination of teams, whence the activation time chromosome is computed; eventually, only the time chromosomes are kept in the population. 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.

The main blocks of the final solving architecture are depicted in Figure 2.3.

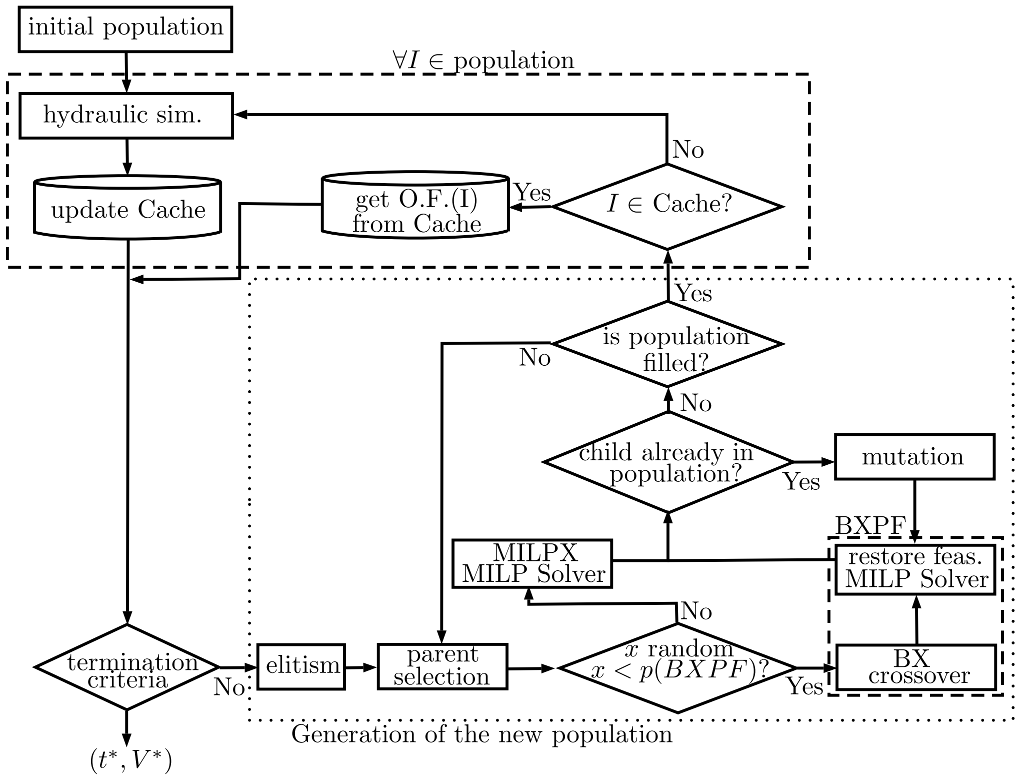


Figure 2.3: Architecture of the time-based Hybrid Genetic Algorithm

2.7 Computational Results

We applied the presented GA s to the water distribution network of Ferrara, Italy, population 130,000. A sketch of the network is reported in Figure 2.4. 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

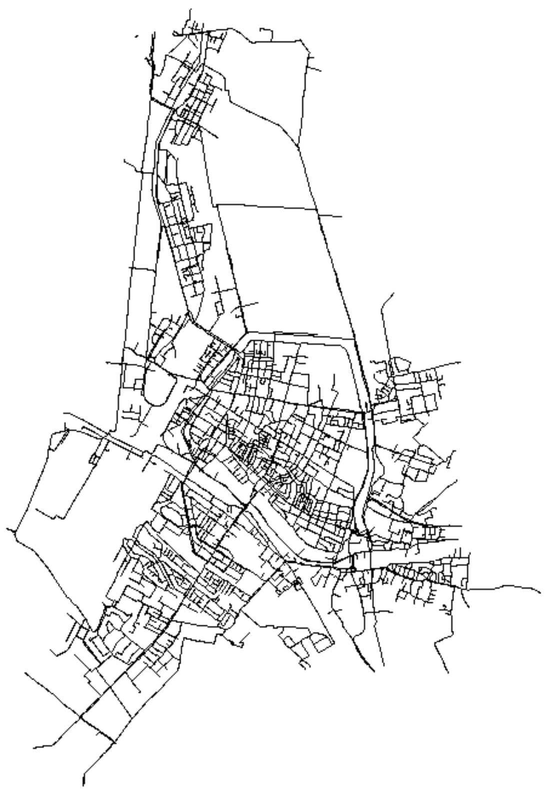


Figure 2.4: Picture of the hydraulic distribution network of Ferrara

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. Travel times range from 5 to 15 minutes, and the maximum pause duration U is set equal to 5'.

A previous work on the same network [15] selected the set of devices to be operated after contamination detection by way of a multi-criteria GA, targeting both minimal number of devices and minimal volume of consumed contaminated water, supposing to have as many teams as devices, all departing at the same time. From the Pareto front provided in [15], 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 [15], 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 [39], 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. This scheduling, in turn, is obtained by solving a MILP model for the mTSP with constraints (2.7a), (2.7c-2.7h), 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*. This study selects 20 scenarios, including the previous 5, selected according to the same criterion.

With respect to [39], we improved the quality of the MIP solver used to tackle the optimization problems in the Hybrid GA s and to compute the minimum makespan schedule. CBC COIN-OR [60] is the open-source code MILP Solver used in [39], while in this study we adopted a commercial MILP Solver, namely Gurobi [61]. 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 [18]. 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 GA s. The first 10 belong to the time-based Hybrid GA s family (section 2.6) 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 GA s belong to the permutation-based family (sections 2.4 and 2.5), namely, $2C^{CS}$, $2C^{VS}$, $2P^{CS}$, and $2P^{VS}$ GA s. For each scenario, we run each GA 100 times. Each GA, at each run, shares the same initial population as the other GA s 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; analysing whether the devised solution approach is able to cope with the most challenging version of our problem, that is the variable speed variant.

2.7.1 Dimensioning the population size

The 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 2.5. The average over 100 runs of the best solution value at each EPANET call is reported, for population size in $\{10, 20, 40, 60, 80, 100\}$. 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 that was indicated by the hydraulic engineers. 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. However, as confirmed in the GA literature, small population sizes affect diversity and may accelerate convergence to medium quality solutions and hamper the search. This can be observed when comparing the evolution of the population size 10 with the population size 20.

The ideal population size may also depend on the specific representation to be used. Since in this case it was calibrated with respect to $2C^{CS}$ representation, which in turn is sufficiently similar to the $2P$ representation to extend the results, the time based GAs might benefit from an ad hoc calibration of the population size. Keeping this in mind in the further analysis we believe that our final conclusions may only further benefit from this possibility. Figure 2.5 suggests in fact to set $N_{pop} = 20$.

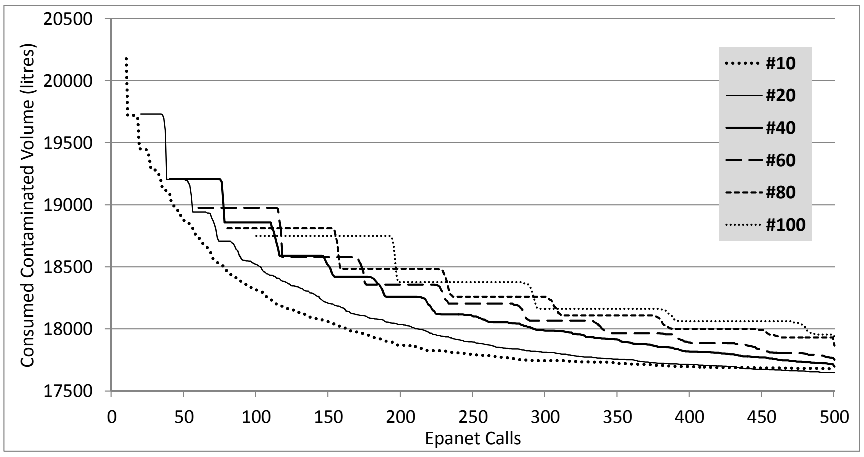


Figure 2.5: Population size calibration, average on 100 runs of the best solution value of $2C^{CS}$ at each EPANET call, scenario A, 500 simulation calls

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 other studies such as [55] where there is no such a tight limitations on the number of function evaluations, calibration yielded much smaller values than 100 such as 40.

2.7.2 Impact of high performance solvers

As a consequence of the bottleneck step represented by the simulation time and of the limit on their number, the running time allowed to each algorithm is basically the same, and most important, due to the caching mechanism, the number of different solutions each algorithm inspects is also equal, 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. Figure 2.6 shows the distribution of the running time of the Gurobi MILP solver on 9000 runs. 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.

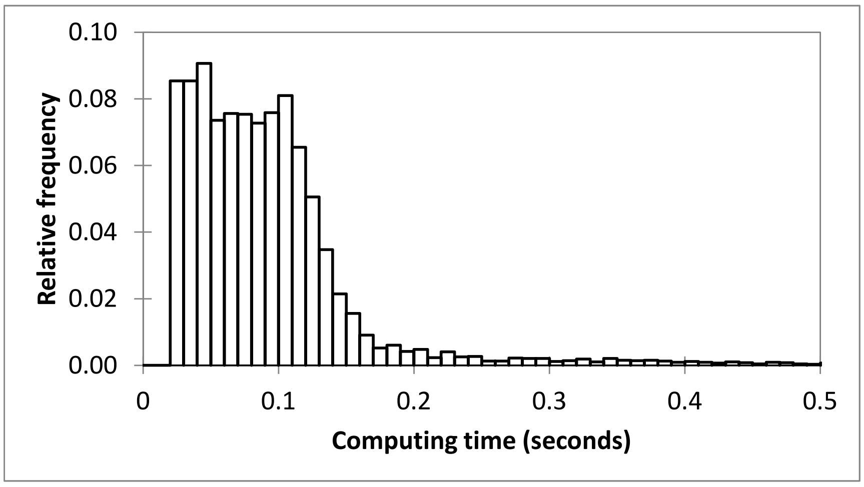


Figure 2.6: MILP solver running times distribution over 9000 runs

Let us discuss pros and cons of using high performance MILP solvers in our hybrid GA s. 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 regard an increasing number of clones being generated by $MILPX$. In order to discuss these issues in further details let us analyse the data of 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 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 2.7 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. In the case of constant speed, an absolute tolerance of 10 reduces to one half, on average, the percentage of calls devoted to feasibility restore, and a tolerance of

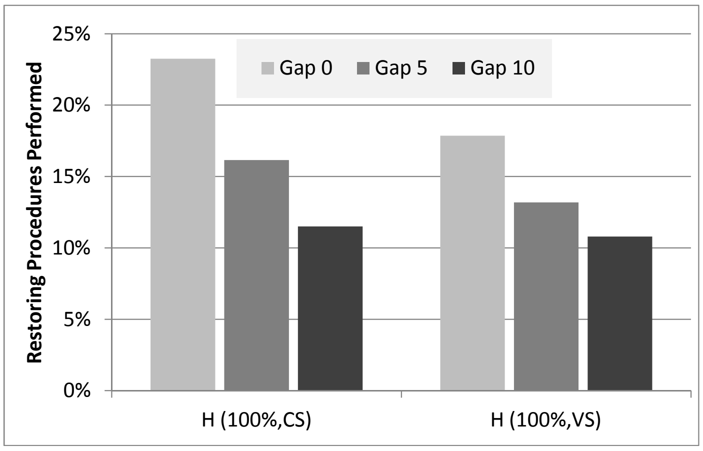


Figure 2.7: Impact of tolerance gap on the number of feasibility restore after mutations.

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, so the solver simply identifies it, but it does not have to 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. The degree of randomness of the choice is controlled by the tolerance parameter, which defines the distance of those vertices that are considered to be adequately acceptable. 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 between the pure $MILPX$ and $BXPF$.

Table 2.1 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

Table 2.1: 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

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, this effect is more evident for the variable speed case, in which there is a higher number of feasible solutions within a given distance from a vertex.

The different behaviour in case of constant or variable speed may be due to the following reason. The variable speed feasible region is much wider than the constant speed one, and it is also locally compact. Therefore, also the set of the feasible values for the operation time of a device is much wider, and, there are probably several feasible points in the neighbourhood 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.

2.7.3 General comparison of the proposed variants

Figure 2.8 provides an overall picture of the quality of the results of the different GA s, 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 best performing encoding for the mTSP, i.e.,

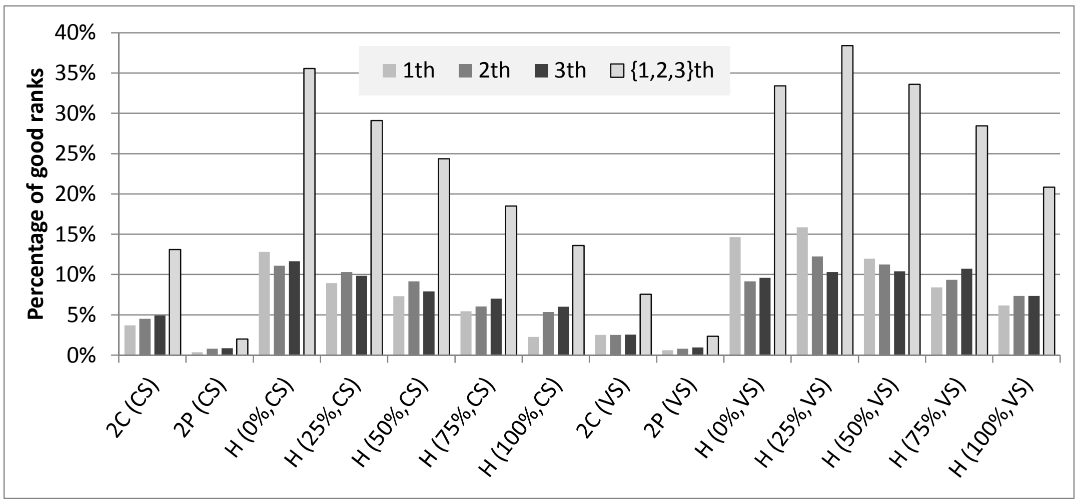


Figure 2.8: Rankings 100×20 runs

$2P^{CS}$, 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.

Note that it is worth facing the challenge of the variable speed variant. The best performing algorithm is indeed $H(25\%, VS)$. Moreover, any hybrid variable speed GA improves on any permutation based GA.

The impact of the relative presence of the two crossovers in the hybrid GA s 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 previous observation that the variable speed variant of $MILPX$ is less affected by the phenomenon of clones.

Statistical analysis

In order to have a scientific comparison of 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 dataset.

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:

- 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;
- 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 GA starts 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* [62, 63].

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* [64] 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 the literature, the most known (and the most conservative) method is the *Bonferroni correction* [65, 66], which defines the new confidence level as $\alpha^* = \alpha/Number\ of\ comparisons)$.

All tests have been carried out with XLSTAT, a commercial Microsoft Excel add-on [67]. A useful reference textbook for nonparametric statistical analysis is [63].

All the Friedman tests performed on the 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 GA s over the whole set of scenario, we first performed the Nemenyi test procedure on each consistent dataset, and then, for each possible comparison, we computed the number of times in which a comparison is statistically significant (given the opportune correction) over the 20 scenarios.

To ease presentation, we split the statistical analysis into two blocks: in the first we compare the sequence-based algorithms with the activation time-based algorithms, while in the second, we compare the activation time-based amongst them.

In the first, the aim is to show if, in general, the Hybrid GA s outperform the others based on sequences, both for the constant and the variable speed. Table 2.2a and Figure 2.9a show the percentage of significant comparisons between each GA with $2C$ and $2P$ GA s in the cases of constant speed, while Table 2.2b and Figure 2.9b are for the case of variable speed. 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 2.9a shows that the Hybrid GA loses effectiveness increasing the $MILPX$ selection chance.

In the second, we performed the Nemenyi procedure only over the datasets related to the Hybrid algorithm. Tables 2.3a and 2.3b report, as the previous tables, the percentage of significant comparisons of the tested configurations over the set of 20 scenarios, and an overview is given by Figures 2.10a and 2.10b. There, it can be seen that for the CS Hybrid GA a lower frequency of $MILPX$ corresponds to an efficiency loss, see for example the percentage of significant comparisons of $H(0\%, CS)$ and $H(25\%, CS)$ w.r.t. $H(75\%, CS)$ and $H(100\%, CS)$ in Figure 2.10a. This is not observed for the variable speed case (Figure 2.10a), where for all $MILPX$ configurations the percentage of significant comparisons lies under the 50% of cases.

The boxplots in Appendix 3.8.3 provide a schematic view of the performance of the 14 GA s for each of the 20 scenarios, for all the 100 runs. The specific scenario affects the performance of all the GA s. In fact, the variance of the results of each GA varies on the different scenarios according to same pattern; for example, all GA s have a high variance on scenario P and a much lower one on scenario T.

We noticed that the GA s 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, it follows that the activation of the n devices is not equally effective in each scenario. Moreover, we can observe that in the worst scenarios, namely I and P, the variable speed variant hybrid GA s are not competitive with respect to the ones at constant speed. We argue that this might be due to the choice of the

Table 2.2: Nemenyi Post Hoc Analysis: for each pair of algorithms we show the percentage of scenarios in which the algorithm in row performs better than the algorithm in column, with a statistical significance lower than $\alpha^* = 0.05/21 = 0.0024$

(a) Constant Speed

	$2C^{CS}$	**$2P^{CS}$**
$2C^{CS}$	-	90%
$2P^{CS}$	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%

(b) Variable Speed

	$2C^{VS}$	**$2P^{VS}$**
$2C^{VS}$	-	20%
$2P^{VS}$	0%	-
H (VS,0%)	100%	100%
H (VS,25%)	100%	100%
H (VS,50%)	95%	100%
H (VS,75%)	85%	100%
H (VS,100%)	80%	90%

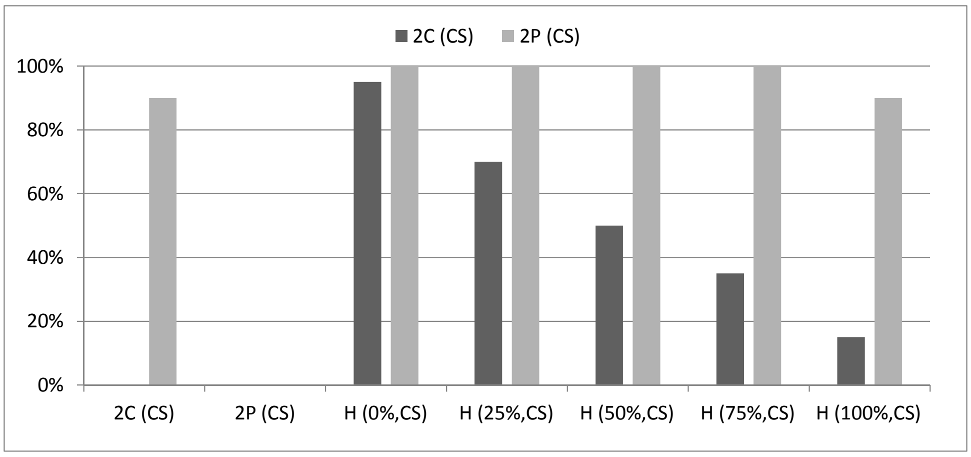
Table 2.3: Nemenyi Post Hoc Analysis: for each pair of algorithms we show the percentage of scenarios in which the algorithm in row performs better than the algorithm in column, with a statistical significance lower than $\alpha^* = 0.05/10 = 0.005$

(a) Constant Speed

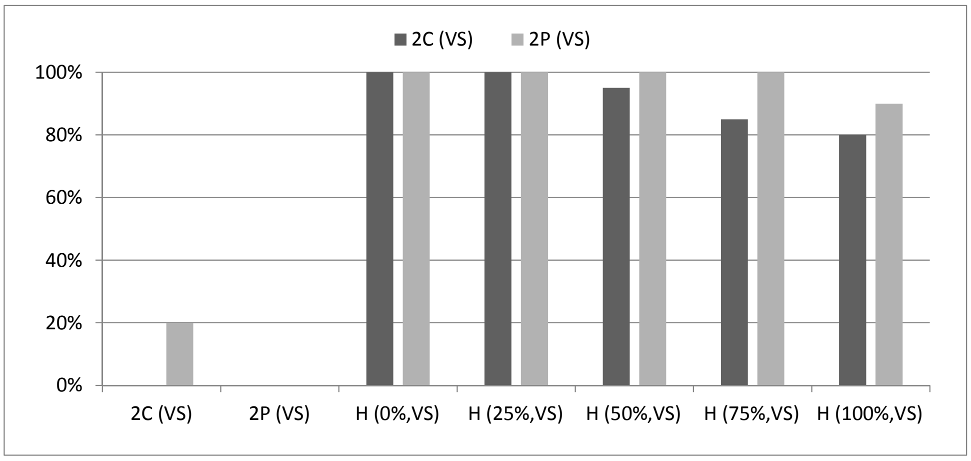
CS	**H(0%)**	**H(25%)**	**H(50%)**	**H(75%)**	**H(100%)**
H(0%)	-	5%	50%	65%	80%
H(25%)	0%	-	15%	50%	85%
H(50%)	0%	0%	-	5%	50%
H(75%)	0%	0%	0%	-	10%
H(100%)	0%	0%	0%	0%	-

(b) Variable Speed

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

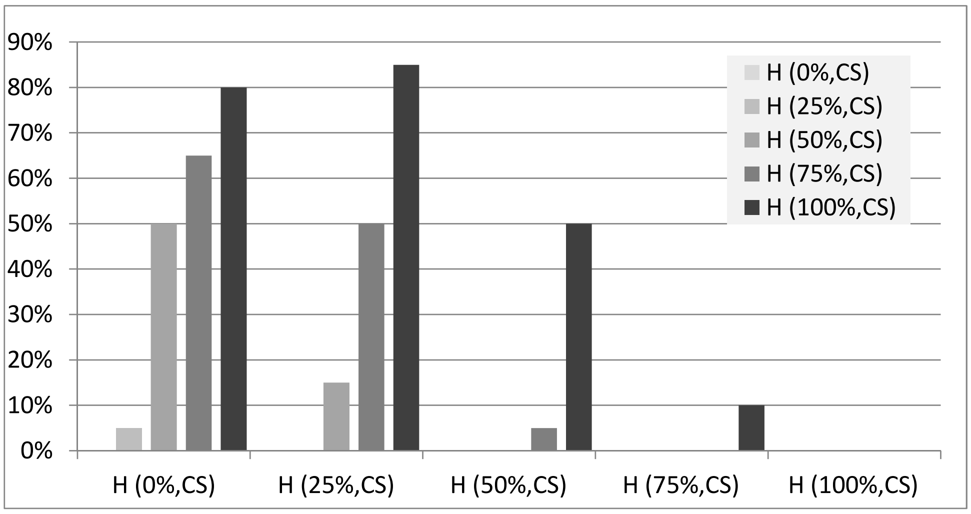


(a) Constant Speed (Table 2.2a)

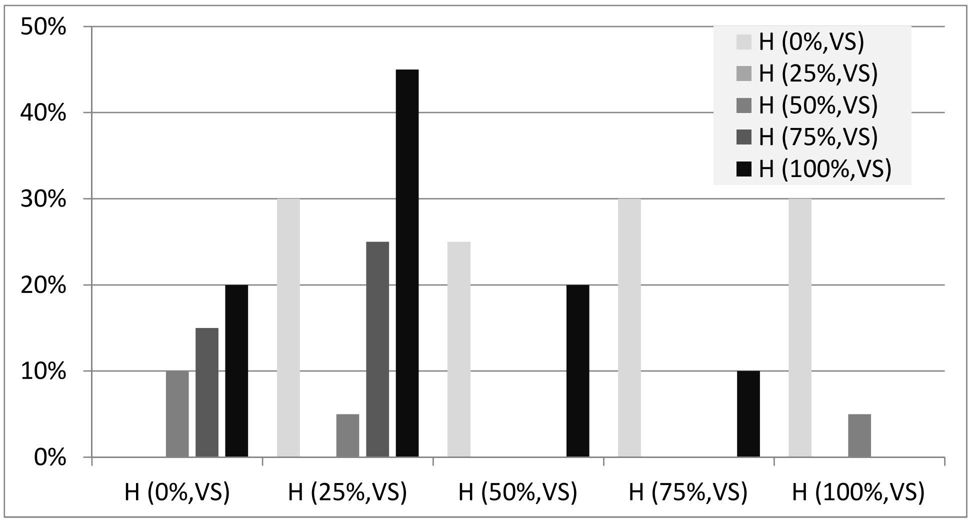


(b) Variable Speed (Table 2.2b)

Figure 2.9: Nemenyi Post Hoc Analysis (from Tables 2.2a and 2.2b)



(a) Constant Speed (Table 2.3a)



(b) Variable Speed (Table 2.3b)

Figure 2.10: Nemenyi Post Hoc Analysis (from Tables 2.3a and 2.3b)

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.

Comparison with common sense and local search approaches

Moreover, we want to provide computational evidence to support our intuition regarding the inefficacy of common sense inspired solutions, such as those minimizing the *makespan* of the activation process as well as the *latency* of device activations. To this purpose we implemented and solved the corresponding MILP models. These are basically minor variants of the model presented in 2.6.1, differing in the objective function. The maximum and the sum of the $t_i^{\mathcal{F}}$ is minimized in the makespan and in the latency case, respectively.

A second comparison involves the use of blind random search (BRS). In this case, the GAs performance is compared against the results provided by the inspection of 500 randomly chosen feasible solutions. We implemented a BRS for both the constant and variable speed cases, and used the same procedure adopted for the initial population generation.

A further investigation concerns the supposed poor performance of neighbourhood based heuristics when such a limited number of function evaluations is allowed. To support this intuition, we implemented a classical Local Search (LS) for the constant speed case. The neighbourhood is given by all solutions obtained by moving a device from its current position to any other feasible location. In order to make a wise use of the limited number of function evaluations, we adopted a first improvement scheme and avoided to visit solutions already inspected. A maximum of 500 simulation is allowed to each LS. The starting solution is randomly chosen according to the same procedure used to build the initial population of the GAs. For a fair comparison, the LS was run several times, each time starting from a different solution. As expected, the search trajectory of LS visits on average a small number of points, namely 17, and the solutions inspected are all very similar to each other, which makes the returned solution much dependent from the starting solution. A local minimum is reached within the 500 evaluations in 38% of runs and the average number of calls is 461. This is not surprising considering that the local optimality proof itself requires $O(n^2)$ evaluations and it suggests that not much improvement could by gained by a more sophisticated neighbourhood based metaheuristics able to escape from local minima. In our case LS keeps the search confined into a limited subset of the feasible region whereas it could be possibly used as an intensification procedure to be applied to elite solutions returned by the GAs.

Figure 2.11 provides a global picture of the performance of alternative methods (i.e., latency, makespan, BRS-VS, BRS-CS and LS) when compared to the best performing GAs ($H(CS, 0\%)$ and $H(VS, 25\%)$). The average solution values are

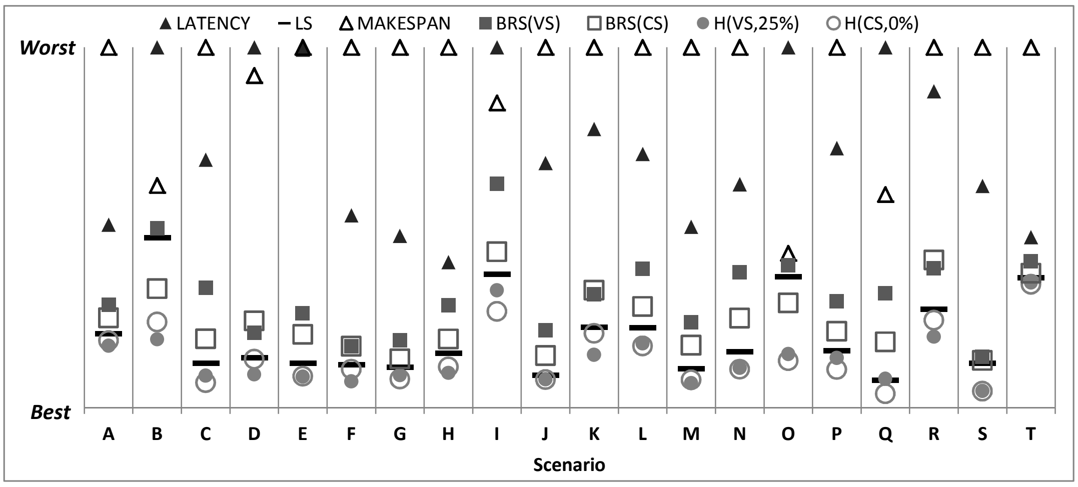


Figure 2.11: Percentage position of the averaged best solutions found by each method within the best and the worst solutions

reported for BRS-VS, BRS-CS, LS, and the two GAs. For each scenario, values are scaled as follows to enhance readability. Making 0 the value of the best solution ever computed and making 100 the value of the worst among all solutions, each solution value is rescaled in this range. It can be noted that GAs always perform better than BRSs; the LS is actually competitive only for 7 of the 20 selected scenarios and its generalization to the variable speed case is not straightforward.

Comparison with previous works

The best 6 solutions that are known for the Ferrara's network were computed by GAs in [15], and are:

- a schedule where devices are activated instantaneously after the alarm;
- 5 "handmade" schedules with $N_{teams} = \{1, 2, 3, 4, 7\}$.

The schedules were evaluated by EPANET over 42 contamination scenarios and the averaged volume of contaminated water consumed by the users were in order: 50,725, 52,511, 44,799, 44,571, 44,287, and 47,287 litres. The best schedule was then obtained by considering the availability of 4 teams. Every GA proposed hereby computes better schedules with only 3 teams. For example, any configuration of the hybrid GA is able to find solutions with averaged volume lower than $34,000$ litres, so much better than the ones in [15].

2.8 Intensification by Path Relinking

Despite the hybrid genetic algorithm based on activation times (Section 2.6) computes on avarege better solutions than the other approaches, it lacks of robustness for some of the tested contamination scenarios (see Section 2.7). For those scenarios the boxplots shown in Appendix 3.8.3 report large boxes for common GAs as well as the hybrid ones (see for example boxplot for scenario K); so, even though the hybrid GAs provide on average good solutions the variance may be high. This is due to the fact that GAs often get stuck on local optima, around the starting population.

Also, notice that the experimental analysis performed on each scenario consists of 100 runs, which means about 70 hours of computing time. For real applications, even offline, it would be necessary to consider a lower number of calls.

All these motivations are incentives to research further on improving robustness of the proposed GAs. For example, it might be possible to perform an *intensification* step just on the final population found by any GA run, but this would not be very effective, in fact by definition a final population of a genetic algorithm contains a high similarity, even more whenever the algorithm is stuck in a local optimum. In such a case additional EPANET calls would be then used during the exploration of a search space which is not promising anymore. A promising approach would be to exploit the dataset in output to different GA runs, i.e., a bunch of best populations.

Path Relinking (PR) [68] is a well known intensification technique. Working on a *reference set* (rs) composed of several solutions, PR first selects from rs a reference (r) and a target (g) solution, then it iterates valid moves to transform step by step r into g (Figure 2.12). This procedure allows for the exploration of the path between two good solutions, according to the hypothesis that a better one can be found among the feasible solutions in the middle. In literature the reference and the target are also called “initial” and “guiding” solutions respectively.

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

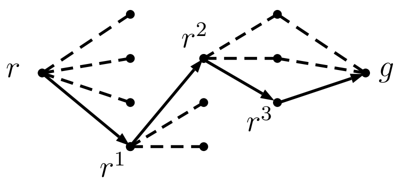


Figure 2.12: Graphical representation of Path Relinking

The building blocks of a Path Relinking algorithm are:

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

In our case, the bunch of best populations given by some GA runs provides naturally the reference set of PR, the target g can be easily selected as the best solution in rs, and the reference r has to be selected properly through quality and diversity criteria, as Section 2.8.1 reports. In Section 2.8.2, taking the cue from several aspects of the genetic encodings proposed in Sections 2.4 and 2.6, the neighbourhood structure is discussed.

This is the first time Path Relinking is used to address the mTSP, but in the last decade it has been applied to many variants of its generalization, i.e., the Vehicle Routing Problem (VRP) [59, 69]; several studies combine in fact PR to Taboo Search [70], GRASP [71–73], and co-operative search [74].

Moreover, PR has been already coupled to Genetic Algorithms for several problems [75–77], but this is the first application combining GA and PR together on optimisation problems in hydroinformatics and more especially on vehicle routing variants.

Finally, the only approach exploiting path relinking in a simulation-optimisation context was proposed to address a resource allocation problem [78] in oil industry, it combines Particle Swarm Optimisation and PR.

2.8.1 Selection of reference candidates

As stated before, the reference set can be built up from the final populations of the GAs. In particular, both quality and diversity from the target have to be taken into account; thus, different metrics can be combined together to filter properly the initial dataset.

The distance between two solutions can be evaluated considering the routes of the teams as well as the activation times. Despite in the former studies the diversity is measured on the graph representation of the VRP [70–74], in this case the preferred way is to measure the diversity over the time representation. In fact the graph representations of the solution would introduce a huge amount of redundancy (see Section 2.4), this means that the same vector of activation times can be mapped into different trees that may differ a lot w.r.t. the metrics defined for graph representations.

The metrics here proposed for the time representation are the Hamming distance

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

where $d_i = 1$ if $g_i \neq r_i$ and 0 otherwise, and the Euclidean distance

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

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

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

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

2.8.2 Solution representations

In order to solve this mTSP variant by Path Relinking a proper solution representation is necessary.

The sequence based variant

As stated before, for Vehicle Routing several PR approaches have been already proposed in the literature; since mTSP and VRP have some problem structures in common (e.g., the routes), those studies can provide a good starting point to design a suitable solution representation.

In particular, the neighbourhood structure for VRP problems is built up from the subset of *customers* that are served by different routes [70], the new position of a customer can be then computed following cost driven decisions [70] or considering the shortest path to transform a route of r into a route of g [73]; a memory can be used to store information that can help the search in the next steps, and, finally the reference set can be updated dynamically during the search [74]. In these studies the solutions are represented as strings of symbols, in which any route is expressed by the set of visited customers; in this representation the *relocate* operator is massively adopted and adapted to each particular VRP variant.

A similar representation was also used to define the "sequence" genetic encoding and it has been extensively discussed in Section 2.4: the routes are explicitly defined by strings of devices and the order within each string defines the visiting order of the team. In our mTSP variant there is no analytical formalization of the subtour's cost and several VRP constraints are relaxed, thus the neighbourhood structure is much easier and it can be drawn through the predecessors of the hydraulic devices. In the i-th iteration of the PR procedure, let $p^i_r(dev)$ and $p_t(dev)$ be the predecessors of dev in r^i and g respectively, the set $P^i = \{dev \in \{1..N_{dev}\} \mid p^i_r(dev) \neq p_t(dev)\}$ be the set of devices of r^i having different predecessors in g. The neighbourhood at the step i is then defined as the solutions r^{i+1} in which for at least one $dev \in P^i$ the property $dev \notin P^{i+1}$ must hold. In other words, the way to move from r^i to r^{i+1} is to fix into r^i one predecessor according to g. On the tree representation this move "relocates" one or more devices into the same route or across different routes, making the reference tree closer to the target. Whenever $P^i = \emptyset$ the target is reached and the algorithm ends up. Since $P^{i+1} \subset P^i$ the algorithm always ends in a finite number of steps.

The whole neighbourhood at the step i consist of all those devices having a different predecessor in t. Given the current device i its whole neighbourhood is explored and evaluated by EPANET;

The predecessors for solutions shown in Figure 2.13 are listed in Table 2.4. Here r and g share only the predecessor of 3, so $P^0 = \{1, 2, 4, 5, 6, 7, 8\}$ and its cardinality ($|P^0|$) is 7. A valid move would be for example to relocate 5 after its predecessor in g, i.e., 3. Let 2 be the best move after the 7 EPANET calls, then in r^1 2 is relocated after 5. The tree representation of r^1 is shown in Figure 2.13, now it shares two equal predecessors with g and the neighbourhood size is 6. The next step is to move from r^1 to r^2, having $P^1 = \{1, 4, 5, 6, 7, 8\}$; now, moving only 5 without preserving the subroute "5–2" would be destructive w.r.t. the previous

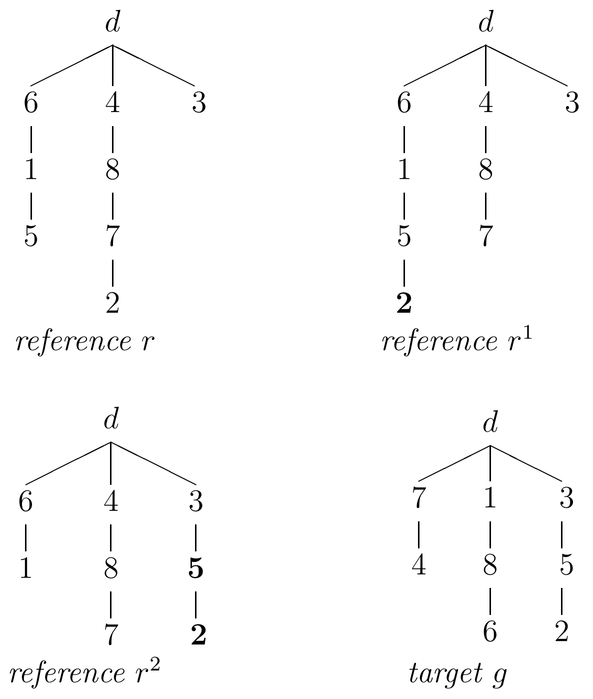


Figure 2.13: Feasible References and Target solutions for 8 devices

Table 2.4: Predecessors list of Figure 2.13

dev	r	r^1	r^2	g
1	6	6	6	d
2	7	5^*	5^*	5
3	d	d	d	d
4	d	d	d	7
5	1	1	3^*	3
6	d	d	d	1
7	8	8	8	d
8	4	4	4	6

moves. Consequently, to keep valid the previous moves the algorithm must use a memory, where the previous choices are stored; in Table 2.4 the devices having a fixed predecessor are highlighted by a $*$. The use of this memory also allows for the relocation of subroutes; in fact, moving 5 will now assign the subroute 5–2 to the device 3.

Moreover, since empty routes are considered as infeasible solutions, the algorithm prevents the move to empty a route.

This version of PR moves at most N_{dev} times and calls EPANET at most $\frac{N_{dev}(N_{dev}+1)}{2}$ times.

From now, we refer to this version as the “sequence” PR ($PRseq$).

The time based variant

Similarly to the time based encoding described in Section 2.6, the time based variant of the PR works on the activation times of the solutions. In such a case r and g are vectors of times and the i-th neighbourhood is given by the set of feasible vectors having at least i elements in common with g and the others as similar to the r^i ones as possible. It is clear that a feasible vector with these characteristics always exists, namely the target vector.

This neighbourhood structure can be represented in a more practical way by the indexes of the elements of r^i that differ from g, i.e., $P^i = \{k \in [1..N_{dev}] \mid r^i_k \neq g_k\}$. Unfortunately, in this representation assigning $r^i_k = t_k$ may yield an infeasible time vector.

The MILP program (2.7a)-(2.7h) can be used with the purpose to compute the closest feasible vector to r^i having $r^i_k = g_k$ for the choosen k; furthermore, in order to keep valid the previous moves, every previous assignment has to be imposed in the current step. In other words, at the i-th iteration the input to the solver would be: r, g, the assignments $r^j_h = g_h$ from the previous iterations $j < i$, the current assignment $r^i_k = g_k$. This should be done for every $k \in P^i$, which means that the feasible moves are at most $|P^i|$; the best one w.r.t. the EPANET evaluation is chosen. Whenever the neighbourhood collapses into the target solution, i.e., every $k \in P^i$ leads to the target vector, the PR ends. Figure 2.14 shows an example of this: at the first iteration the best move was with $k = 5$, at the second with $k = 1$; the current iteration is now the 3-th, and a new call to the MILP solver is going to be performed with the new $k = 8$. Notice that the previous assignments are passed again to the solver.

Since the number of feasible moves decreases at each iteration of at least one unit, the maximum number of EPANET calls is again $\frac{N_{dev}(N_{dev}+1)}{2}$.

Notice that, as the Genetic Algorithm based on activation times, this PR is

$$r^3 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline \mathbf{13} & & & & \mathbf{43} & & & \mathbf{31} \\ \hline\end{array}$$

$$\uparrow r_1^2 = g_1 \qquad \uparrow r_5^1 = g_5 \quad \uparrow r_8^3 = g_8$$

$$g = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 13 & 24 & 33 & 25 & 43 & 19 & 37 & 31 \\ \hline\end{array}$$

Figure 2.14: Input to the MILP solver at the i-th iteration of the PR

also hybrid, thus we will refer to it as the "hybrid" PR (PRh).

This MILP model could be also extended in order to compute only the feasible vector r^{i+1} having at least one more element in common with g, with the hasty attempt to save precious EPANET calls. In such a case at most N_{dev} points would be explored and this would limit too much the exploration capabilities of the PR algorithm.

The final architecture

The two variants share the same algorithm architecture, shown in Figure 2.15. The "MILP solver" module is enabled only in the PR based on activation times. In this schemata the generic element of P^i is named e and it is considered to be either a device for the $PRseq$ or a time vector's index for the PRh. Notice that also the PR algorithm involves a cache containing the volumes of the solutions already visited. This is very useful because PR may visit, and it often does, identical solutions; in this way, PR is restarted on several references until the maximum number of simulations is reached.

The proposed general architecture to optimise the countermeasures in case of contamination events in a water distribution system is depicted in Figure 2.16. As shown here, GAs and PR blocks share the same cache, so a solution is not evaluated twice along the solving blocks.

2.8.3 Computational results

The experimental analysis is based on the assumption that having a limited number of EPANET calls it would be better to consume a big part of them in performing hybrid genetic algorithms, in order to visit as many regions of the search space as possible; then, Path Relinking is used to explore the paths among those regions, looking for better solutions.

The hypothesis is that allotting $n - 1$ parts of hydraulic simulations to GAs plus the last part to the PR is better than spending everything into GAs, even more on those scenarios where GAs lack of robustness.

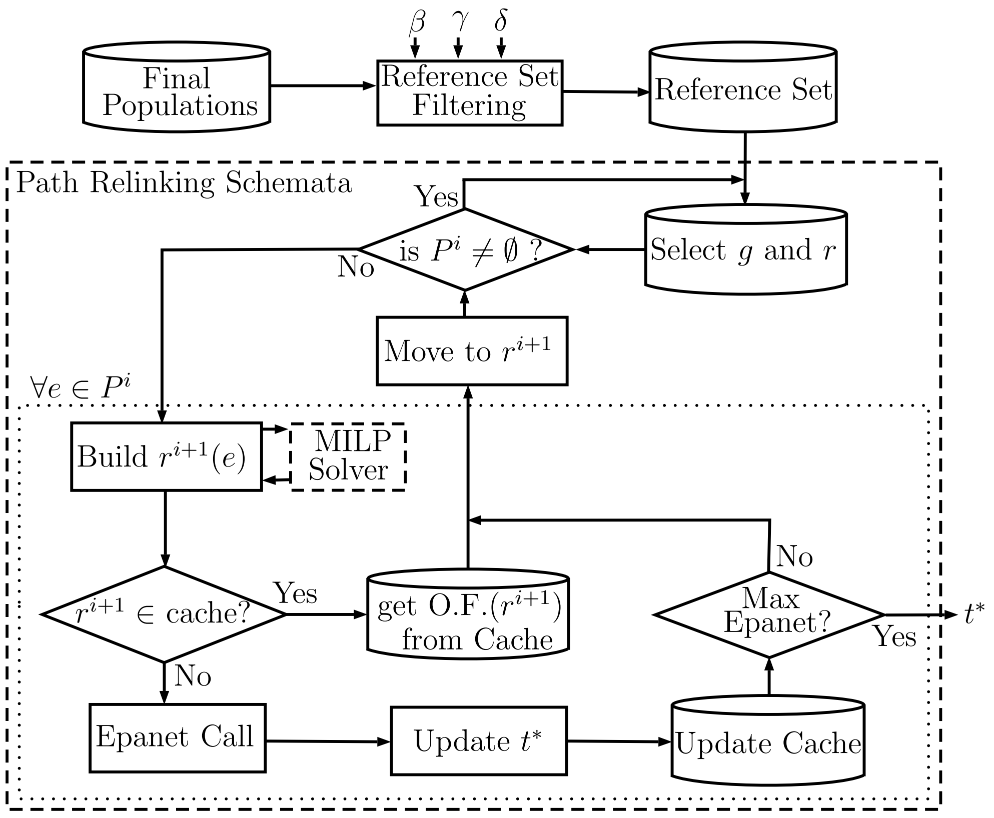


Figure 2.15: Algorithm architecture of Path Relinking

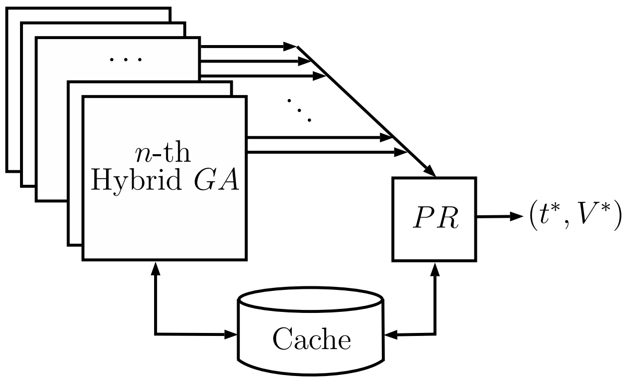


Figure 2.16: Global solving architecture

The reference set was built up from the final populations of 10 GA runs (every GA terminates after 500 EPANET calls). On every scenario, to perform 10 independent runs of PR, 100 GAs have been ran and the final populations have been collected and grouped into 10 groups of 10 final populations; $PRseq$ and PRh have been tested on each of these groups with a limit to the EPANET calls 500 (as any GA).

Table 2.5 reports the number of improvements within 10 runs of $PRseq$ and PRh, also it reports the averaged percentage improvement in terms of volume of contaminated water (for each run the percentage improvement is given as $\frac{V(t^*)-V(t)}{V(t)} * 100$). The rows are sorted by considering the third and the last column

Table 2.5: Number of improvements and average improvement of volume of PR on 10 runs

Scen	♯ IMP		AVE IMP	
	PRseq	*PRh*	*PRseq*	*PRh*
A	8	**10**	**1.21%**	0.81%
K	4	**10**	0.29%	**0.55%**
B	0	**9**	0.00%	**0.30%**
F	4	**8**	**0.45%**	0.42%
C	3	**8**	0.04%	**0.38%**
G	0	**8**	0.00%	**0.38%**
J	5	**8**	0.17%	**0.34%**
D	5	**7**	**0.38%**	0.35%
E	5	**7**	**0.49%**	0.31%
O	0	**7**	0.00%	**0.20%**
Q	6	**7**	0.14%	0.14%
N	3	**6**	0.09%	**0.27%**
T	2	**6**	0.03%	**0.20%**
P	2	**6**	0.02%	**0.15%**
S	2	**5**	0.01%	**0.28%**
I	2	**5**	0.02%	**0.06%**
R	4	**4**	0.11%	**0.15%**
H	3	**4**	**0.23%**	0.10%
M	2	**3**	0.03%	**0.11%**
L	0	**3**	0.00%	**0.03%**

as first and second key respectively. This order clearly exhibits that for some scenarios PR is in general a good intensification approach, e.g., scenarios A, K,

B, F, C, G, and J for which PRh computes a better solution for up the 80% of runs. Moreover, the first three of the above scenarios have a gap between the best and worst solution found by 100 GA runs of about the 15% (see boxplots in Appendix 3.8.3). For other scenarios, namely L, M, H, R, I, and S only the PRh comes up to the 50% of improving runs. In particular, among these scenarios both the PR variants obtain the worst percentage improvement on the L and I; here the gap between the worst and best solutions found by 100 GAs runs is about the 3%, so very tight. It is also clear that PRh always outperforms $PRseq$ in terms of number of improving runs, in every row of ♯IMP the PRh's entry is greater that the $PRseq$'s one indeed. $PRseq$ outperforms PRh only on five scenarios in terms of averaged volume of contaminated water (AVE IMP in the table). Another remarkable result is that allotting every computational resource to the GAs did not outperform the PR, in fact one more independent GA run was not able to find any better solution than the PR.

In general, the gain in terms of quality is up to 1%. This means that the volume of contaminated water would decrease of about 14 litres for scenario T and 144 litres for scenario B; these might seem insignificant amounts but even a single glass of toxic water can irreparably damage a human life.

2.9 Future improvements

The hypothesis that 10 GAs plus 1 PR is better than 11 independent GAs has been experimented. The results suggest that PR is an appropriate tool to implement an intensification phase. This will be also compared to other intensification procedures, like a GA that starts from the final population of the previous GA runs. Statistical analysis should validate the results.

Last experiments on PR algorithms have shown that intensification procedures can help whenever GAs get stuck on local optima. This paves the way towards a new hybrid algorithm, joining capabilities of genetic algorithms, mathematical programming, and path relinking in a more integrated architecture. However, the experimental investigation on PR effectiveness was performed by considering the hybrid GAs as a first independent step (Section 40), and the question now would become whether PR could be fully integrated into it.

One idea would be to allot an amount of the available hydraulic simulations to several small genetic algorithms; before the complete convergence of GA modules the algorithms allocates the computational resources left to the path relinking procedures. This approach needs to be finely tuned on the number of GAs, the size of population, the number of generations, and then the amount of computational resources to assign to the path relinking step. Also the structure of the architecture has to be thoroughly designed, in fact GA and PR steps can be organized as chains or rings. In the former case the educated populations would be the reference set of an intensification procedure, then the bunch of solutions found by PR could be reinserted into the initial population of a second genetic algorithm. In the latter case one path relinking module does an intensification on a pair of educated populations coming up from parallel genetic algorithms, then a final intensification can be performed on the former PR solutions.

Furthermore, the PR literature proposes several strategies to select the target and the reference solution. Also, the target can change dynamically during the search, whenever a new best candidate is reached. All these opportunities should be properly integrated and tested.

Chapter 3

Optimal placement of Isolation Valves

In this chapter we introduce a real problem in hydraulic engineering concerning the location of the isolation valves of a Water Distribution System (WDS), and reformulate it as a graph based optimization problem.

Isolation valves are closed whenever broken pipes need to be isolated and fixed by technicians; this entails a service disruption for the users. Also, apart installation, valves lead to huge maintenance costs, because they are very vulnerable to failures. This limits the isolation system to a number of devices and their positioning must be thoroughly designed in order to:

- guarantee a good resilience of the hydraulic network;
- minimize costs of the isolation system;
- minimize the unsatisfied demand of the users during the isolations.

Locating the isolation valves is part of the design issues of the urban hydraulic networks, and it arises whenever a city district is either planned from scratch or renewed. Typically, depending on the demographic growth of a region, a design phase is expected to take place every 30-100 years in developed and developing countries, and it takes long time and huge financial efforts. Thus, together with other design issues, the positioning of isolation valves comes up as a strategic optimisation problem, because it has a bearing on long-term financial assets, network resilience and users satisfaction. In Section 3.1.1 the problem is described more in detail and its main structures are detected and highlighted.

Until a few years ago the location of hydraulic devices such as valves was planned following rules of thumb and the experience of the engineers. In the last two decades, several techniques in computer science have been exploited to set up non-exact approaches. For example, being genetic algorithms fully integrated into MATLAB [17] and other technical tools, many studies rely on this technique in hydraulic literature. Nevertheless, no mathematical models have been yet proposed by hydraulic engineers for this optimisation problem, but we claim it as a necessary step in order to understand complexity and combinatorial aspects of the problem. After formal and mathematical definitions and exact solving procedures are settled, new non-exact approaches can be studied and developed in a more aware fashion. Refer to Section 3.2 for a detailed literature review about previous approaches.

This Chapter wants to unify the studies in [79–81], where mathematical models in Mixed Integer Linear Programming (MILP) (Section 3.4) and two logic programming formulations are proposed (Section 3.5.1). Whenever possible a comparison between the two paradigm in terms of declarativeness and effectiveness will be provided w.r.t. the Bottleneck Isolation Valves Location Problem (BIVLP). Computational results are presented in Section 3.7. Furthermore, a novel Benders decomposition is proposed in Section 3.8.1. Future perspectives are finally discussed in Section 3.8.

3.1 Problem definition

3.1.1 Valves closure and sector isolation

WDSs are complex systems whose mission is to supply water to the communities living in their service area. A WDS is made of several components, the main ones being: a set of reservoirs feeding the WDS, a set of pipes delivering water to the system users, a user demand for each pipe, describing the average water consumption by the users served by that pipe (litres per second $[l/s]$), a set of junctions each one describing the connection of two or more pipes to each other. We illustrate these components on the toy network depicted in Fig. 3.1. This hydraulic network has a single reservoir T, 8 junctions and 10 pipes with positive demand, plus a 0-demand pipe which connects the reservoir to the rest of the network.

Users are connected to their closest pipe by way of smaller pipes through which water is supplied. To this purpose, we can imagine users as being evenly distributed along the pipe. In turn, pipes receive water at an adequate pressure from the reservoirs to which they are connected in the hydraulic network. Usually, the topological layout of hydraulic networks contains a few loops that increase network reliability. Thus, a pipe can be connected to a reservoir by several different

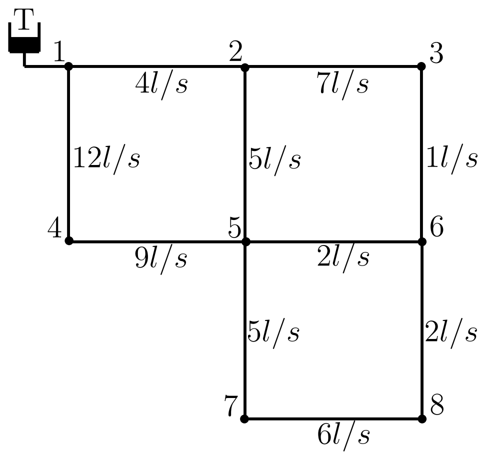


Figure 3.1: A simple hydraulic network

paths. Given a pipe, if each path from the pipe to each reservoir is interrupted (or closed) by any valve, water pressure falls, the pipe no longer supplies its users and it is said to be *isolated.* On the contrary, if any *open path* from any reservoir exists the pipe is considered to be fed. Failure of ageing pipes frequently occurs. In such a case, the leaking pipe is isolated on purpose, to be dewatered and fixed. Isolation is achieved by closing some of the isolation valves purposely located on the network, in such a way that the failed pipe gets disconnected from the reservoirs. In an ideal situation, each pipe would have one such valve positioned at each of its two extremes, so that only that pipe could be disconnected in case of maintenance by closing just its two valves, and it would require twice as many valves as the network pipes. However, the number of valves is limited due to cost, and their location poses a challenge, as described hereafter.

First, valves must be properly located at pipe extremes, right in adjacency to the junctions; in fact, manholes are typically available there to make the junctions accessible for maintenance purposes. Also, every pipe can get broken, thus any pipe must be isolable closing some valves. Consequently, when all valves are closed the network is subdivided into a set of subnets (or *connected components* in graph theory).

Definition In hydraulic networks, the subnets that are induced by closing all valves are called *sectors.* The valves that delimit a sector s are colled *boundary valves* of s.

Said in another way, a sector is a set of pipes which are always connected, even when all valves are closed. It follows that pipes within the same sector share the same status, either isolated or fed by a reservoir, depending on which valves are closed. When a sector is isolated, all its users experience supply disruption. Figure 3.2 reports a feasible isolation system made of 7 valves, where $v_{a,b}$ tells that the valve lies on junction a of the generic pipe (a, b); similarly $v_{b,a}$ tells that valves is on the extreme b pipe (a, b). This feasible positioning consists of: $v_{1,2}$, near junction 1 on pipe $(1, 2)$; $v_{1,4}$, near junction 1 on pipe $(1, 4)$; $v_{2,3}$, near junction 2 on pipe $(2, 3)$; $v_{3,6}$, near junction 3 on pipe $(3, 6)$; $v_{5,4}$, near junction 5 on pipe $(5, 4)$; $v_{6,8}$, near junction 6 on pipe $(6, 8)$; and finally $v_{7,5}$ near junction 7 on pipe $(7, 5)$. Four sectors are induced by those valves, namely $s_1 = \{(1, 2),(2, 5),(3, 6),(5, 6),(5, 7)\}$ with internal demand $ID(s_1) = 17l/s$; $s_2 = \{(1, 4),(4, 5)\}$ with $ID(s_2) = 21l/s$; $s_3 = \{(2, 3)\}$ with $ID(s_3) = 7l/s$; and finally $s_4 = \{(6, 8),(7, 8)\}$ with $ID(s_4) = 8l/s$. Notice that in this configuration s_1 is the biggest sector in terms of internal demand. The boundary valves of s_1 are $v_{1,2}$, $v_{2,3}$, $v_{3,6}$, $v_{5,4}$, $v_{5,7}$, and $v_{6,8}$; for s_2 are $v_{1,4}$ and $v_{5,4}$; for s_3 are $v_{2,3}$ and $v_{3,6}$; and for s_4 are $v_{5,7}$ and $v_{6,8}$.

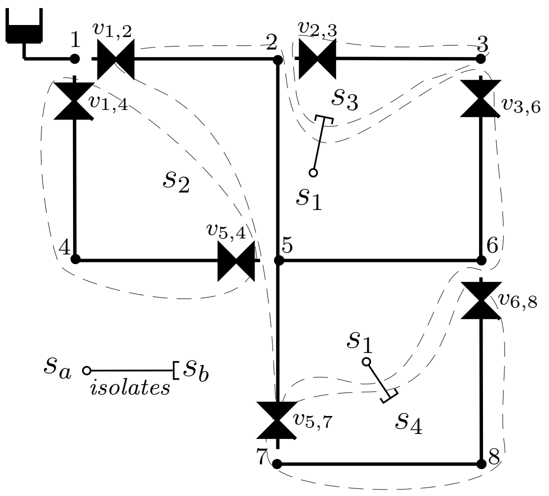


Figure 3.2: A feasible isolation system for the net in Figure 3.1

As far as the quality of the solution, the WDS engineers who design the network aim to reduce and equally distribute the service disruption among users in case of maintenance operations. Graph Partitioning Problem (GPP) recall several aspects of this problem structure, and Section 3.3 deepens this analogy, turning out with a suitable graph representation of the problem.

However, a secondary effect of sector isolation called *unintended isolation*

Table 3.1: Sector specifications for Figure 3.2

Sector	Pipes	Demand $[l/s]$	ID $[l/s]$	Unintended Isolations	UD $[l/s]$
s_1	$(1,2)$	4	17	s_3,s_4	32
	$(2,5)$	5			
	$(3,6)$	1			
	$(5,6)$	2			
	$(5,7)$	5			
s_2	$(1,4)$	12	21	none	21
	$(4,5)$	9			
s_3	$(2,3)$	7	7	none	7
s_4	$(6,8)$	2	8	none	8
	$(7,8)$	6			

makes Graph Partitioning unable to measure properly the total unsatisfied demand due to sectors isolations. In fact, due to unintended isolations the unsatisfied demand $UD(s)$ when s is isolated might be greater than $ID(s)$.

3.1.2 Unintended Isolations

A pipe for which all connections to the reservoirs go through the isolated sector will be isolated as well when that sector is closed. Therefore, the supply disruption associated to a sector cannot take into account only the internal user demand of the sector itself, but must consider the demand of unintentionally isolated pipes as well. Having at hand Table 3.1 and Figure 3.2 pipes $(2,3)$, $(6,8)$ and $(7,8)$ are pointedly isolated whenever a pipe in s_1, e.g., $(5,6)$ gets broken. Closing s_1's boundary valves also determines the isolation of s_3 and s_4, so the latter are unintended isolations of s_1, and $UD(s_1) = ID(s_1) + ID(s_4) + ID(s_4) = 17 + 8 + 7 = 32l/s$ is now the worst unsatisfied demand. In this example the role of unintended isolation is pretty clear, being $ID(s_1) < UD(s_1)$; generalizing this into the equation

$$ID(s) \leq UD(s) \tag{3.1}$$

the quality measure of this problem oversteps from the internal demand of sector, which is the common metric for GPP. To include the missing quantity and achieve the entire unsatisfied demand of a sector isolation this particular problem structure should be modelled.

Recall that fed pipes are the ones having at least one open path to a source, and contrariwise every path from sources to isolated pipes is interrupted by some closed valve; these facts can be exploited by those technologies and declarative languages for which the concept of *path* is easily captured: it is actually the case of Logic Programming, as described in Section 3.5.

Furthermore, this feature deeply relies on *water flows*, being flows the amounts of satisfied demands of users. Network flows are modelled in Mathematical Programming by means of conservation and capacity equations, avoiding the possible exponential growth of paths on a highly connected graph. To handle unintended isolations over a main GPP formulation a proper graph representation and a mathematical model are presented in Section 3.4.2.

3.1.3 The Isolation Valves Location Problem

The Isolation Valves Location Problem (IVLP) of WDSs consists of locating a limited number of valves at pipe extremes so that any pipe can be isolated. What an optimal placement is may depend on several criteria that give rise to different objective functions and then to different problem specializations; in particular, the Bottleneck Isolation Valves Location Problem (BIVLP) minimizes the maximum undelivered demand given a number of valves [79]. In this definition the probability of failure is the same for any pipe and the cost of the positioning is not considered within the objective function. Here Eq. (3.1) is extended as

$$\max_s\{ID(s)\} \leq \max_s\{UD(s)\},$$

meaning that the maximum internal demand can be lower than the maximum unsatisfied demand varying the isolated sector s. Accordingly, the objective function can be stated in a general fashion as

$$\min : \max_s\{UD(s)\}.$$

3.2 Related works

The two main issues related to the IVLP problem addressed in the hydraulic engineering literature are: the identification of the *segments* (i.e., the sectors) and unintended isolation due to the closure of some isolation valves, and the optimal location of isolation valves. Regarding the first topic, among others, [82,83] exploit a dual representation of the network, with segments treated as nodes and valves as links; [7, 8] exploit the topological incidence matrices to identify the segments. Regarding the second topic, both [7] and [8] tackle the problem by bi-objective genetic algorithms, seeking a compromise between cost and solution quality. In particular, the former minimizes the number of valves and the maximum supply disruption. The latter minimizes the cost of the installed valves related to pipe diameters and the average supply disruption weighted by the probability of pipe failure. Both provide an approximation of the Pareto frontier and no bounds to evaluate the quality of the heuristic solutions.

The first exact approach for the IVLP we are aware of is based on Constraint Logic Programming on Finite Domain (CLP(FD)) [27], and models the problem as a two-players game and three moves: player 1 locates the valves, player 2 chooses a pipe to break, and player 1 closes a set of valves [34]. Here sector identification is an integrated part of the search strategy and it consists of a graph visit from the broken pipe towards the closest reachable valves.

A recent exact approach models stochastic pipe failures [84], and it is based on the mathematical models developed in this work. To find feasible solutions within reasonable computing time it also proposes a heuristic solving procedure; this decomposes the problem into two layers: the hard one that involves the positioning of valves and is partially solved by a greedy heuristic, whereas the easier layer completes the solution by solving several simplified stochastic subproblems. Although the model has been decomposed, the resulting decomposition schema is not solved by following the classical iterative procedure, where at each step the subproblems pass useful information to be used in the next upper-level step.

The valve placement problem has some similarities with the graph partitioning problem, in which the goal is to partition a graph into (almost) equal-size subgraphs. Graph partitioning is NP-hard, and most works deal with heuristics or approximation algorithms. A pioneering work [85] proposes a greedy algorithm which iteratively improves a k-way partition of a graph by solving a 2-way partition on different pairs of node subsets. A different approach is based on the *spectral* analysis of the graph [86]. Other related problems are the multicut problems [87], in which the aim is to remove the minimal set of edges such that given pairs of nodes are no longer connected. Despite these similarities, the valve placement problem can not be tackled by the solution approaches that have been developed for the graph partitioning problem and for the multicut problem, none of which can handle the issue of unintended isolation. However, it describes the feasible region of the IVLP and consequently it has a fundamental role in this problem; next section tightens this relationship.

3.3 Hydraulic sectors and graph partitioning

The Graph partitioning is one of the most studied problems in combinatorial optimization, and admits several variants. Recall that a partition of a set is a collection of non-empty disjoint subsets whose union returns the set. Generally speaking, GPP consists of partitioning the vertices of a graph into a set of connected components by removing a minimum (weight) set of edges, according to some criteria.

Definition The set of removed edges to disconnect the components is referred to as a *multicut.*

Often, the number of components of the partition is fixed, and the number of nodes in each component is bounded from above. Literature references are many, among which [88] investigates the polytopes associated to the integer programming formulations of the major variants, and [89] studies the convex hull of incidence vectors of feasible multicuts for the capacitated GPP.

The several analogies between the hydraulic network sectorisation and graph partitioning, as recently exploited in the hydraulic engineering literature [90], seem at first to provide an ideal framework for modelling BIVLP. Recall that each pipe must belong to one sector in order to be isolated if required, but only one sector, due to the minimality of sector definition. Therefore, sectors induce a partition on the positive demand pipes, whose associated multicut models the location of the isolation valves at pipe extremes. As stated before, this simple correspondence, though, does not allow us to address the issue of unintended isolation. To meet this requirement in MILP, we need to introduce an extended graph that represents the topology of the network, on which "flow variables" will model water flows, thus capturing unintended supply disruption. We first provide a graph representation of he hydraulic network supporting graph partition, and in the Section 3.4.2 we show how to extend such graph to handle unintended isolation.

First, we model the user demand of a pipe (i, j) as concentrated in a single point denoted ϵ_{ij}, located in the middle of the pipe. Denote by δ_{ij} such demand. Then, we shrink all the reservoirs into a single one, to which we associate node σ, and connect this super–reservoir to the junctions that were originally connected to the sources (as shown in Figure 3.3).

Now we can introduce the graph G that represents the junctions and pipes of the hydraulic network as nodes and edges, respectively.

Definition The undirected and weighted graph $G = (V, E, W)$ consists of:

V the set of verteces that is union of:

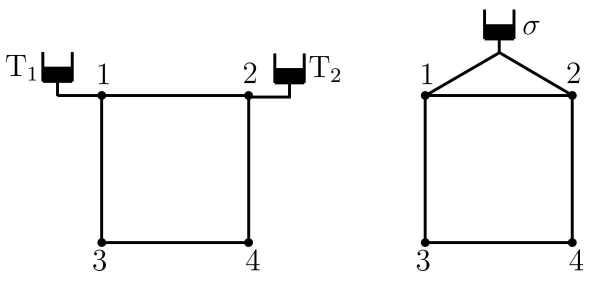


Figure 3.3: The introduction and connection of the super–reservoir σ

σ the reservoir (or tank)

J the set of nodes modelling the junctions of the hydraulic network

E the set of edges that is union of:

T the set of the structural edges corresponding to 0-demand pipes which connect σ to a vertex in J;

Ψ the set of the edges modelling the pipes.

W the set of weights containing the demands δ_{ij} for any $(i, j) \in E$.

Fig. 3.4 shows how the two adjacent pipes of the hydraulic network are modelled in graph G.

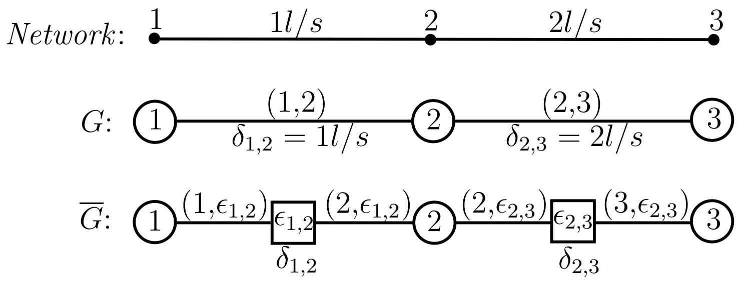


Figure 3.4: Two adjacent pipes of the hydraulic network and their representation in G and $\overline{G}$.

In this graph representation the multicut would consist of pipes; but to disconnect an entire pipe two valves should be placed on the its extremes. In this application the pipe can host only one valve, so the multicut should consist of the "pipe's extremes". In other words, in the IVLP the cut passes through the valves rather than on the pipes. Thus, the extension of G should represent all possible places can host a valve. Since a pipe can host two valves on its extremes and weights on GPP are typically on nodes rather than on edges, the new graph $\overline{G}$ represents pipes as weighted nodes (say ϵ nodes) and connects each pipe-node to its two junction-nodes by means of undirected edges.

Definition The extended undirected graph $\overline{G} = (\overline{V}, \overline{E}, \overline{W})$ is made of:

$\overline{V}$ the set of vertices, s.t. $\overline{V} = V \cup \Gamma$, and $\Gamma = \{\epsilon_{ij} \mid \ (i,j) \in E\}$;

$\overline{E}$ the set of edges, s.t. $\overline{E} = T \cup \overline{\Psi}$, and $\overline{\Psi} = \{(i, \epsilon_{ij}), (j, \epsilon_{ij}) \ \mid \epsilon_{i,j} \in \Gamma\}$;

$\overline{W}$ the set of weights associated to the vertices in $\overline{V}$, s.t. $\overline{W} = \{w_u \mid u \in \overline{V}\}$ and

$$w_u = \begin{cases} \delta_u & \text{if } u \in \Gamma \\ 0 & \text{if } u \notin \Gamma \end{cases}$$

Since now weights are only on the ϵ–nodes, in this graph δ_{ij} will be also referred to as $\delta_{\epsilon_{ij}}$.

Each edge in $\overline{\Psi}$ can host a valve, while edges in T do not. In real life WDSs, actually, a special valve is always present on each pipe in T in order to isolate the reservoir from the network, if the reservoir needs maintenance. However, such a valve would never be closed for pipe maintenance purposes, so we disregard it here. Therefore, we assume that the edges in T bear no valves. Furthermore, to keep notation simple, we suppose that all pipes other than those incident on the reservoirs have positive demand.

The MILP model we propose in this work needs to be dimensioned in the number of sectors and valves that are allowed.

Definition The number of available valves is N_v, and the maximum number of sectors admitted is N_s.

Recall that $UD(s)$ is the undelivered demand associated to sector s, for $s \in S = \{1, .., N_s\}$, and it is given by the sum of δ_{ij} for each pipe (i,j) which gets isolated when the boundary valves of sector s are closed. We search for the location of at most N_v valves on as many edges of $\overline{\Psi}$ yielding at most N_s sectors such that $\max_{s \in S}\{UD(s)\}$ is minimum. The mathematical and logic programs we propose for the GPP structure of BIVLP, Section 3.4 and 3.5.2 respectively, need to be dimensioned on N_s; in fact some variables and constraints of the mathematical model and some predicates of the logic program rely on an explicit indexing of the sectors. Consequently, the relation between N_v and N_s has a role on the effectiveness of the solving procedure; Section 3.6.1 delves into this aspect. Moreover, explicating the sectors' names may yield to *symmetries*; this is also a valid reason to:

- make the bound on N_s as tight as possible (Section 3.6.1);
- find good strategies to break the symmetries (Section 3.6.2).

Moreover, different configurations of valves may yield the same cost of the sectors; this should be prevented through symmetry breaking strategies (Section 3.6.1). Valves can be also redundant, in the sense that they may not contribute to the multicut; to avoid this the cycles of the graph can be exploited, as described in Section 3.6.4.

3.4 Mathematical models for the Bottleneck Isolation Valves Location Problem

In Operations Research (OR) and applied mathematics Constrained Optimisation Problems (COPs) can be often stated in MILP [24], which is a mathematical paradigm. MILP programs can be solved by using different techniques, the most used combines a Branch and Bound (B&B) search algorithm and the famous Simplex algorithm [24]. Some of the most known solvers are: Coin-OR CBC [60], CPLEX [91], Gurobi [61], SCIP [92], XPress [93] and others. MILP has been widely used to address graph partitioning and network design problems [88,89,94].

This section first describes a graph partitioning model on graph $\overline{G}$. How to extend $\overline{G}$ to model unintended undelivered demand and finally to address the BIVLP is described in Section 3.4.2. A *bilevel* [95] MILP model for the IVLP is presented in Section 3.4.4. In Section 3.4.5 the bilevel model is then reduced into a single level MILP by exploiting the *min cut – max flow* theorem [96].

3.4.1 Modelling network sectorization

To model the sectorization of the network in $\overline{G}$ we introduce the following sets of variable:

$\tau_{ij}^{s} \in \{0,1\}, \forall\ (i, \epsilon_{ij}) \in \overline{\Psi}, \forall\ s \in S$, is a binary variable equal to 1 if a boundary valve for s is located on pipe (i, j) near i in the hydraulic network (edge (i, ϵ_{ij}) in $\overline{G}$), and 0 otherwise.
Likewise, $\tau_{ji}^{s} = 1$ if the valve is near j, that is, on edge (j, ϵ_{ij}) in $\overline{G}$.

$z_i^s \in \{0,1\}, \forall\ i \in \overline{V}, \forall\ s \in S$, is a binary variable equal to 1 if node i belongs to sector s and 0 otherwise.

The following constraints partition the vertices of $\overline{G}$ into at most N_s sectors with limited user demand δ_{max}, by cutting at most N_v edges in $\overline{E}$.

$$\sum_{s\in S} z_i^s = 1, \qquad \forall\ i \in \overline{V} \tag{3.2a}$$

$$\sum_{s\in S}\sum_{\epsilon_{ij}\in\Gamma} (\tau_{ij}^s + \tau_{ji}^s) \leq 2N_v \tag{3.2b}$$

$$\sum_{\epsilon_{ij}\in\Gamma} z_{\epsilon_{ij}}^s \delta_{\epsilon_{ij}} \leq \delta_{max}, \qquad \forall\ s \in S, \tag{3.2c}$$

These constraints impose the following. Each node in $\overline{V}$ belongs to one and only one sector (3.2a). At most N_v valves are available (3.2b); being valves boundary of exactly two sectors the amount of $\tau_{i,j}^s$ summed over $s \in S$ must be reduced by half; this constraint is a reformulation of

$$\sum_{\epsilon_{ij}\in\Gamma} \left(\frac{1}{2}\sum_{s\in S}\tau_{ij}^s + \frac{1}{2}\sum_{s\in S}\tau_{ji}^s \right) \leq N_v$$
$$\sum_{\epsilon_{ij}\in\Gamma} \left(\frac{1}{2}\sum_{s\in S}(\tau_{ij}^s + \tau_{ij}^s) \right) \leq N_v$$
$$\frac{1}{2}\sum_{\epsilon_{ij}\in\Gamma}\sum_{s\in S}(\tau_{ij}^s + \tau_{ij}^s) \leq N_v$$

The sum of user demands of the edges within the same sector is bounded from above by constraint (3.2c) that excludes very unbalanced partitions with sectors with large demand. However, the threshold δ_{max} as well as the parameters N_v and N_s must be carefully set so that a feasible solution exists and no optimal solution is cut off. Note that there is no constraint requiring to use all the available valves, because increasing the number of valves does not necessarily improve the optimal solution value. For example in Figure 3.5, the value of the worst unsatisfied demand is $10l/s$ with $N_v = \{3,4\}$. Moreover, we want to avoid solutions with useless valves, i.e., valves positioned on an edge whose vertices belong to the same sector. For example in Figure 3.6 $v_{1,4}$ and $v_{5,4}$ are boundary valves of s_1; $v_{1,2}$ and $v_{5,4}$ are boundary valves of s_2; whereas the valve $v_{6,3}$ is not boundary of any sector.

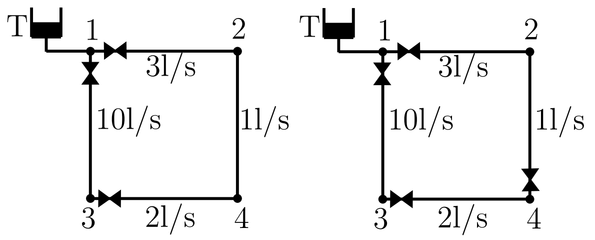


Figure 3.5: Optimal solutions with 3 and 4 valves

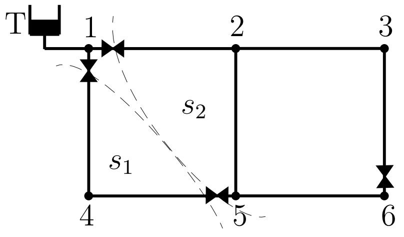


Figure 3.6: Solution with an useless valve

The following constraint states the relation between a valve and the sector of the vertices of the edge where the valve is located, and it imposes that all valves are boundary valves. Recall that τ^s_{ij} refers to a valve positioned between vertex i and vertex ϵ_{ij}. We use the symbol $\oplus$ to denote the XOR logical operator and then provide the system of integer linear inequalities that define the operator.

$$\tau^s_{i,j} = z^s_i \oplus z^s_{\epsilon_{ij}}, \qquad \forall (i, \epsilon_{ij}) \in \overline{\Psi}, \forall s \in S \tag{3.3}$$

Constraints (3.3) state that a boundary valve of sector s is positioned on an edge if and only if exactly one of the two vertices belongs to s. (3.3) can be formulated as the following set of linear inequalities, for each edge $(i, \epsilon_{ij}) \in \overline{\Psi}$ and for each $s \in S$.

$$\tau^s_{ij} \leq z^s_i + z^s_{\epsilon_{ij}} \tag{3.3a}$$

$$\tau^s_{ij} \leq 2 - (z^s_i + z^s_{\epsilon_{ij}}) \tag{3.3b}$$

$$\tau^s_{ij} \geq z^s_i - z^s_{\epsilon_{ij}} \tag{3.3c}$$

$$\tau^s_{ij} \geq z^s_{\epsilon_{ij}} - z^s_i \tag{3.3d}$$

The same constraints hold for any $(j, \epsilon_{ij}) \in \overline{\Psi}$.

Notice that in this formalization also z^s_σ is defined, i.e., also the tank node need to be assigned to a sector. Since we are not interested into the sectorization of node σ and its adjacent nodes we assign them to the additional sector 0, the set of sectors' indexes is now $S = \{0, 1, \ldots, N_s\}$, and the following variables are fixed:

$$z^0_\sigma = 1 \qquad (3.4a)$$
$$z^0_t = 1 \qquad t \mid (\sigma, t) \in T \qquad (3.4b)$$
$$\tau^0_{\sigma j} = 1 \qquad \forall (\sigma, j) \in T \qquad (3.4c)$$

In particular, Eq. (3.4a) imposes the tank nodes to be in the sector 0; Eq. (3.4a) imposes that a valve is installed on every pipe linked to the tank. In order to make the model smaller any other variable and constraints depending on $s = 0$ can be omitted; in fact the other variables in τ^0 and z^0 would be directly set to 0.

Figure 3.7 depicts a piece of solution on a toy network and gives a topological positioning to the variables. Here, any other variable, e.g., $z^2_{\epsilon_{t,a}}$, z^1_b, and $\tau^2_{a,b}$, is 0 in this solution.

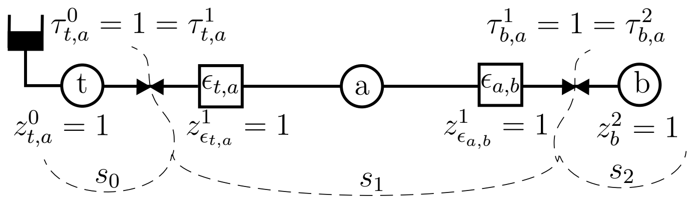

$\tau^0_{t,a} = 1 = \tau^1_{t,a}$
$\tau^1_{b,a} = 1 = \tau^2_{b,a}$
t
$\epsilon_{t,a}$
a
$\epsilon_{a,b}$
b
$z^0_{t,a} = 1$
$z^1_{\epsilon_{t,a}} = 1$
$z^1_{\epsilon_{a,b}} = 1$
$z^2_b = 1$
s_0
s_1
s_2

Figure 3.7: A solution on a small net

The user demand that is unsatisfied when sector s is isolated clearly depends on where the sector's boundary valves have been located, according to the current configuration of the isolation system. Recall that the binary variables that model the positioning of the boundary valves for s are in the set τ^s. We also call τ the entire set of these variable, and it is defined as

$$\tau = \bigcup_{s \in S} \tau^s$$

Since the sector s is drawn by its τ^s, we call:

$ID(\tau^s)$ its internal demand;

$UD(\tau^s)$ the unsatisfied demand due to its isolation;

$SD(\tau^s)$ the satisfied demand when it is isolated.

Reading between the lines, these quantities depend on the variables in τ^s with 1, i.e., the boundary valves of the sector s. In the hydraulic network those valves should be closed to isolate s.

Let us represent the feasible partitioning of the graph drawn by constraints (3.2a-3.2c), (3.3a-3.3d) as $GP(\tau)$. Therefore, BIVLP can be stated as

$$\begin{aligned}
&\min \Delta && && (3.5a)\\
&\quad s.t. && && \\
&\qquad GP(\tau), && && (3.5b)\\
&\qquad \Delta \geq UD(\tau^s), && \forall\ s \in S && (3.5c)
\end{aligned}$$

In other words the optimisation problem in (3.5a–3.5c) minimizes the maximum unsatisfied demand, namely Δ, varying the isolated sector, given a positioning τ of valves that draw a sectorization the network.

In the following, we provide a mathematical description of $UD(\tau^s)$ as the objective value of a nested optimization problem.

3.4.2 Modelling unintended isolations

As stated in the introduction of this chapter the quality of the partitioning must be measured by means of the unsatisfied demand of the users, rather than on the internal demand of the single sector; this is due to the unintended isolations, that extend effects of an isolation towards other portions of the network. In mathematical programming this quantity is can be modelled by considering the satisfied demand; recall in fact that the satisfied demands on an hydraulic network are flows, flowing from the sources to the demand points (i.e., the users). In such a case, given

$$\Upsilon = \sum_{(i,j)\in\ E} \delta_{ij}$$

the total user demand of the network, then

$$UD(\tau^s) = \Upsilon - SD(\tau^s).$$

Basically, this means that the unsatisfied demand when the boundary valves of s (the elements of τ^s equal to 1) are closed amounts to the entire demand of the network but the demand that is satisfied.

The optimisation problem in (3.5a–3.5c) becomes

$$\min \Delta \tag{3.6a}$$

$$s.t.$$

$$GP(\tau), \tag{3.6b}$$

$$\Delta \geq \Upsilon - SD(\tau^s), \qquad \forall\, s \in S \tag{3.6c}$$

The quantity $SD(\tau^s)$ can be described through a flow model on a network when the edges of s have been "deleted"; the flows goes from the source to the demands point, i.e., the ϵ–nodes, through those edges that are still reachable; the ϵ–nodes are connected to a sink that collects all the water flowing in the network (like a sewer). Figure 3.8 gives an insight about how the flows in the small network in Figure 3.6 look like when sector s_2 is isolated. In particular, pipes $(1, 2)$ and $(4, 5)$ are fed, and the water goes from T to the sink P; the other pipes are not fed, being their sector disconnected, and the dashed lines represent empty pipes.

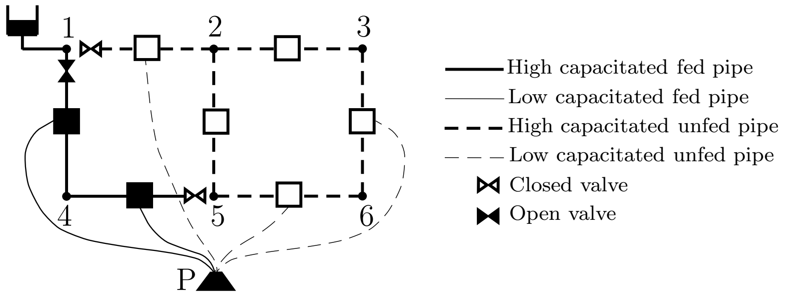

1
2
3
4
5
6
P
High capacitated fed pipe
Low capacitated fed pipe
High capacitated unfed pipe
Low capacitated unfed pipe
Closed valve
Open valve

Figure 3.8: The flows on the network when a sector is isolated

In a flow model the flows have a direction; so the edges of the graph should be oriented. The edges should be also capacitated, in order to limit the flows according with the network capacity. However, graph $\overline{G}$ is not oriented and we can not state a priori where the flows go; moreover, edges in $\overline{G}$ are not capacitated. To model network flows a further extension of the graph is introduced. The idea is to split any edges in Γ, say (i, ϵ_{ij}), into two opposite oriented edges, (i, ϵ_{ij}) and (ϵ_{ij}, i); the same for (j, ϵ_{ij}). Since we have no information about how much water will flow through these related pipes, the edges are high capacitated; however the maximal flow can go out from the tank is the total demand of the network, so these edges have capacity Υ. Also, if a pipe (i, j) is fed, its entire demand is satisfied, this means that between $\epsilon_{i,j}$ and P are flowing $\delta_{\epsilon_{ij}}\, l/s$; since we have not information whether the pipe is actually fed or not the capacity of the new directed edge (ϵ_{ij}, P) is δ_ϵ. Figure 3.8 also shows high and low capacitated edges.

Definition The new oriented graph $\overline{\overline{G}} = (\overline{\overline{V}}, \overline{\overline{E}}, \overline{\overline{W}})$, shown in Figure 3.9, consists of:

$\overline{\overline{V}} = \overline{V} \ \cup \ P$, where P is the *sink*, collecting all user demand that is satisfied;

$\overline{\overline{E}} = \overline{\overline{T}} \cup \overline{\overline{\Psi}} \ \cup \overline{\overline{\Pi}}$ where

- $\overline{\overline{T}} = T$, the set of directed edges going from the sources to the adjacent junctions;
- $\overline{\overline{\Psi}} = \{(i, \epsilon_{ij}), (\epsilon_{ij}, i), (j, \epsilon_{ij}), (\epsilon_{ij}, j) \mid \epsilon_{ij} \in \Gamma\}$, the set of directed edge connecting junctions to the ϵ-nodes and viceversa;
- $\overline{\overline{\Pi}} = \{(\epsilon_{ij}, P) \mid \epsilon_{ij} \in \Gamma\}$, the set of directed edges connecting the ϵ-nodes to the sink;

$\overline{\overline{W}} = \{w_{ij}\}$, the set of costs and that later will be flow capacities, where

$$w_{ij} = \begin{cases} \Upsilon & \forall (i,j) \in \overline{\overline{\Psi}} \\ \delta_{\epsilon_{ij}} & \forall (\epsilon_{ij}, P) \in \overline{\overline{\Pi}} \end{cases}$$

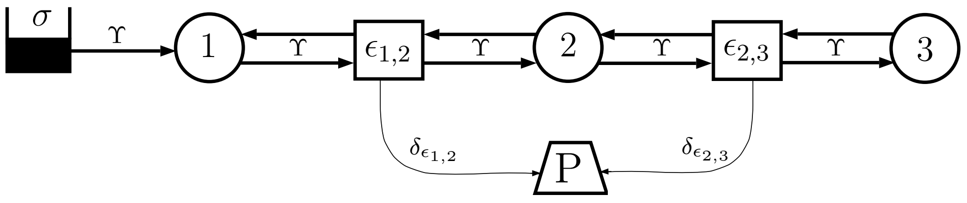


Figure 3.9: The oriented graph $\overline{\overline{G}}$

Let us introduce for this graph representation of the hydraulic network, N_s families of multicommodity flow variables, one for each sector. They are used to represent water flows in the hydraulic network when a given sector is isolated.

First, we define the variables modelling the flow on network pipes. For each $s \in S$ and $(u, v) \in \overline{\overline{\Psi}}$, a pair of *non negative* flow variables are introduced, namely, x^s_{uv} and x^s_{vu}. Recall that $v = \epsilon_{i,j}$ for some pipe (i, j), and $u \in \{i, j\}$. Therefore, each pipe (i, j) yields four flow variables for each sector, namely, $x^s_{i,\epsilon_{ij}}$, $x^s_{\epsilon_{ij},i}$, $x^s_{j,\epsilon_{ij}}$, and $x^s_{\epsilon_{ij},j}$. All such variables are bounded by the sum of users demand Υ, unless a boundary valve for s is located on the edge, say near i. In such a case, $\tau^s_{ij} = 1$ and $x^s_{i,\epsilon_{ij}} = x^s_{\epsilon_{ij},i} = 0$ must hold.

Second, we introduce flow variables on the demand edges in $\overline{\overline{\Pi}}$, connecting each demand vertex ϵ_{ij} to the sink P. For each $s \in S$ and $\epsilon_{ij} \in \Gamma$, let $x^s_{\epsilon_{ij},P}$ be such variable. This variable, for any s, is bounded above by the actual user demand of pipe (i, j), that is $\delta_{\epsilon_{ij}}$ previously introduced.

Finally, for each $s \in S$ and for each edge in $\overline{\overline{T}}$ connecting the reservoir σ to a junction $j \in J$, a non negative flow variable $x^s_{\sigma,j}$ is introduced, with capacity Υ; in fact, those edges can provide at most the total user demand.

In general, in order to compute the delivered demand given a sector isolation, an associated Maximum Flow Problem (MFP) on a graph whose topology depends on the valve placement τ must be solved. In particular, since sectors are isolated one at a time, we can consider separately each such scenario. In order to represent the water flow when a given sector s is isolated, we solve a MFP from the reservoir σ to the sink P with respect to the flow variables indexed by s. For each sector s, the mathematical model of the MFP is defined by *conservation* constraints at the nodes, *capacity* constraints on the flow variables, and the objective function is the maximization of the flow entering in P. Conservation constraints state that the amount of flow in input to the nodes is equal to the amount in output. Capacity constraints set the maximum amount of flow the edges can carry. The optimal value provides the user demand that is satisfied when s is isolated. Capacity goes down to 0 on the arcs where boundary valves of sector s have been located; for example, if $\tau^s_{ij} = 0$ then the flows in (i, ϵ_{ij}) and (ϵ_{ij}, i) are bounded to 0. The problem to maximize the flow in a network whose topology depends on τ^s can be stated in MILP as follows:

$$SD(\tau^s) \quad = max\ x^s_{P,\sigma} \tag{3.7a}$$

$$s.t. \tag{3.7b}$$

$$x^s_{\sigma,i} - \sum_{(i,\epsilon_{ij})\in\overline{\overline{\Psi}}} (x^s_{i,\epsilon_{ij}}) = 0 \qquad \forall(\sigma, i) \in \overline{\overline{T}}, \tag{3.7c}$$

$$x^s_{i,\epsilon_{ij}} + x^s_{j,\epsilon_{ij}} - x^s_{\epsilon_{ij},i} - x^s_{\epsilon_{ij},j} - x^s_{\epsilon_{ij},P} = 0 \qquad \forall\epsilon_{ij} \in \overline{\overline{V}}, \tag{3.7d}$$

$$\sum_{(\epsilon_{ij},P)\in\overline{\overline{\Pi}}} x^s_{\epsilon_{ij},P} - x^s_{P,\sigma} = 0, \tag{3.7e}$$

$$x^s_{P,\sigma} - \sum_{(\sigma,i)\in\overline{\overline{T}}} x^s_{\sigma,i} = 0, \tag{3.7f}$$

$$x^s_{i,\epsilon_{ij}} \leq \Upsilon(1-\tau^s_{ij}) \qquad \forall(i,\epsilon_{ij}) \in \overline{\overline{\Psi}}, \tag{3.7g}$$

$$x^s_{\epsilon_{ij},i} \leq \Upsilon(1-\tau^s_{ij}) \qquad \forall(\epsilon_{ij},i) \in \overline{\overline{\Psi}}, \tag{3.7h}$$

$$x^s_{j,\epsilon_{ij}} \leq \Upsilon(1-\tau^s_{ji}) \qquad \forall(j,\epsilon_{ij}) \in \overline{\overline{\Psi}}, \tag{3.7i}$$

$$x^s_{\epsilon_{ij},j} \leq \Upsilon(1-\tau^s_{ji}) \qquad \forall(\epsilon_{ij},j) \in \overline{\overline{\Psi}}, \tag{3.7j}$$

$$x^s_{\epsilon_{ij},P} \leq \delta_{\epsilon_{ij}} \qquad \forall(\epsilon_{ij},P) \in \overline{\overline{\Pi}}, \tag{3.7k}$$

$$x^s_{\sigma,i} \leq \Upsilon \qquad \forall(\sigma,i) \in \overline{\overline{T}}, \tag{3.7l}$$

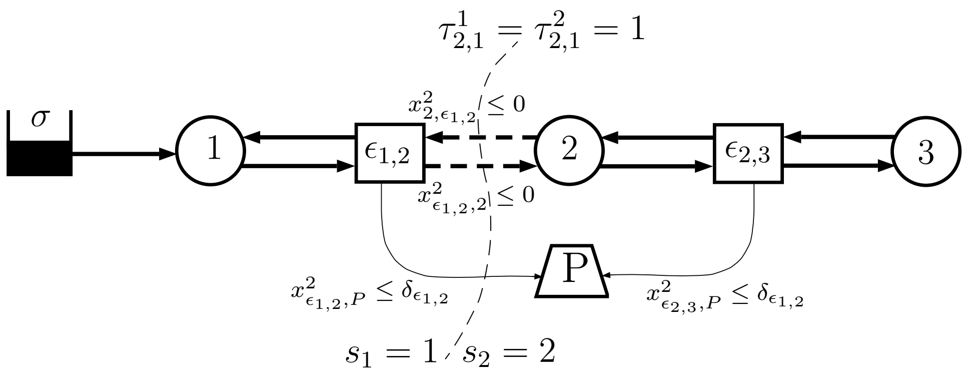
$$x^s_{i,j} \geq 0 \qquad \forall(i,j) \in \overline{\overline{E}}. \tag{3.7m}$$

Constraints (3.7a–3.7m) model the MFP for a given sector s. A fake arc going from the sink P to the source σ with non negative flow variable $x^s_{P\sigma}$ is introduced so that the problem can be stated as a circulation problem. Arc (P,σ) is the only arc outgoing from P and entering σ, therefore the objective function can then be stated as maximizing $x^s_{P,\sigma}$. Conservation constraints at junction nodes are given in (3.7c), at the sink in (3.7d), and at the reservoir in (3.7e), at demand nodes in (3.7f). Capacity constraints for the flow variables on the edges in $\overline{\overline{\Psi}}$, which depends on valves location, are given in (3.7g-3.7j). In particular, if $\tau^s_{ij} = 1$ the flow in (i,ϵ_{ij}) for s is $\Upsilon(1-1) = 0$; contrariwise, if $\tau^s_{ij} = 0$ the flow in (i,ϵ_{ij}) for s is lower than $\Upsilon(1-0) = \Upsilon$. Similarly, capacity constraints for the flow variables defined on the demand edges in $\overline{\overline{\Pi}}$ are given in (3.7k).

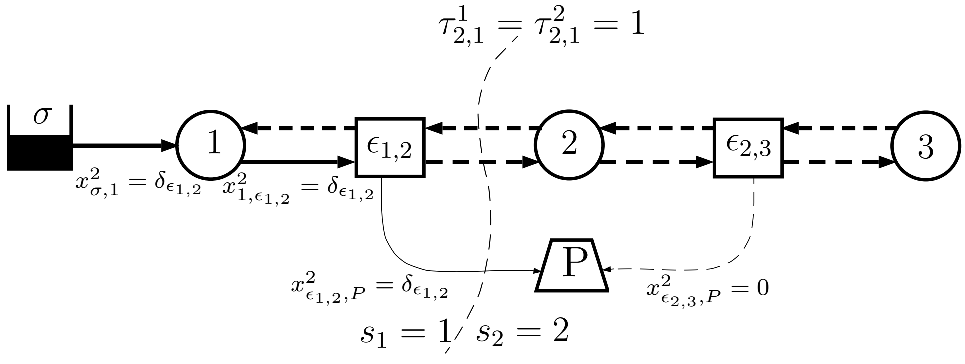
This linear program is a common MFP reformulated on $\overline{\overline{G}}$ but the flows on edges in $\overline{\overline{\Psi}}$, and indirectly those in $\overline{\overline{\Pi}}$ (towards the sink), depend on τ^s, i.e., the boundary valves of s. Remember that $\tau^s = \{0,1\}$ and consider a feasible set of fixed values $\overline{\tau}^s$, this means that the network for sector s is smaller than the original, being some edges 0-capacitated; for example (3.7g) becomes:

$$x^s_{i,\epsilon_{ij}} \leq \begin{cases} \Upsilon(1-0) \leq \Upsilon & \text{if } \overline{\tau}^s_{ij} = 0 \\ \Upsilon(1-1) \leq 0 & \text{otherwise} \end{cases} \qquad \forall\ (i,\epsilon_{ij}) \in\ \overline{\overline{\Psi}}$$

Whenever some flows in $\overline{\overline{\Psi}}$ are forced to be null for the sector s, also some edges in $\overline{\overline{\Pi}}$ are in turn disconnected; Figure 3.10 shows it on the small instance in Figure 3.9. In particular in Figure 3.10a shows capacity constraints for sector 2 when $\tau^2_{2,1} = 1$, and Figure 3.10b shows the maximum flow solution.


$\tau^1_{2,1} = \tau^2_{2,1} = 1$
σ
1
$\epsilon_{1,2}$
2
$\epsilon_{2,3}$
3
$x^2_{2,\epsilon_{1,2}} \leq 0$
$x^2_{\epsilon_{1,2},2} \leq 0$
P
$x^2_{\epsilon_{1,2},P} \leq \delta_{\epsilon_{1,2}}$
$x^2_{\epsilon_{2,3},P} \leq \delta_{\epsilon_{1,2}}$
$s_1 = 1$
$s_2 = 2$

(a) Capacity constraints for sector 2 if $\tau^2_{2,1} = 1$


$\tau^1_{2,1} = \tau^2_{2,1} = 1$
σ
1
$\epsilon_{1,2}$
2
$\epsilon_{2,3}$
3
$x^2_{\sigma,1} = \delta_{\epsilon_{1,2}}$
$x^2_{1,\epsilon_{1,2}} = \delta_{\epsilon_{1,2}}$
P
$x^2_{\epsilon_{1,2},P} = \delta_{\epsilon_{1,2}}$
$x^2_{\epsilon_{2,3},P} = 0$
$s_1 = 1$
$s_2 = 2$

(b) A solution of maximum flow

Figure 3.10: The flow model on $\overline{\overline{G}}$ for sector 2

This MILP model for the MFP has to be defined for any $s \in S$. The flows' capacities depend on variables τ^s, which are meant to be an input of the program. Next section explains how to integrate this model into the main GPP program, achieving a unique MILP model for the BIVLP.

Finally, the resulting model can be generalized, in order to achieve a general framework for the IVLP; Section 3.4.4 describes this reformulation.

Integrating MFPs into GPP

To complete the optimisation model (3.6a–3.6c) the maximum flow must be imposed so that quantity $SD(\tau^s)$ correctly measures the satisfied demand given a sector disconnection; the model can be stated as following:

$$\min \Delta \tag{3.8a}$$

$$s.t.$$

$$GP(\tau), \tag{3.8b}$$

$$\Delta \geq \Upsilon - SD(\tau^s) \qquad \forall\ s \in S, \tag{3.8c}$$

$$MFP(\tau^s) \begin{cases} SD(\tau^s) = max\ x^s_{P,\sigma} \\ \qquad s.t. \\ \qquad\quad FP(\tau^s) \end{cases} \qquad \forall\ s \in S. \tag{3.8d}$$

where $FP(\tau^s)$ is the set of flow constraints (3.7a–3.7m). This formulation can not be solved by any off-the-shelf MILP solver, because there are several maximization statement. Fortunately, in this case minimizing the maximum unsatisfied demand means basically to maximize the minimum satisfied demand. This can be also obtained mathematically by considering that Υ is a constant, and

$$\min_s \{\Upsilon - SD(\tau^s)\} = \max_s \{SD(\tau^s)\}.$$

Consequently, the main objective is now to maximize Δ, and the optimisation statements within the submodels (3.8d) can be omitted:

$$\max \Delta \tag{3.9a}$$

$$s.t.$$

$$GP(\tau), \tag{3.9b}$$

$$\Delta \leq x^s_{P,\sigma} \qquad \forall\ s \in S, \tag{3.9c}$$

$$FP(\tau^s) \qquad \forall\ s \in S. \tag{3.9d}$$

Notice that this model holds because we are interested into maximizing only the satisfied demand of the worst sector, say $\dot{s}$; in fact, in the optimal solution of this model, there is no guarantee that the flows of other sectors $s \neq \dot{s}$ are maximal. Whenever the objective function of the general IVLP needs to guarantee the maximality of every sector's flow, this model will not fit anymore. In Section 3.4.4 this model is generalized in order to achieve the maximality of any sector's flow.

The model in (3.9a–3.9d) can be passed to a common off-the-shelf MILP solver. Next section analyses the possible weaknesses a common Branch and Bound resolution may suffer of.

Finally, the formulation in (3.8a–3.8d) is still very useful, being the basis to define a suitable optimisation model for the general IVLP, as described in Section 3.4.4.

3.4.3 Weaknesses of common Branch and Bound

Continuous relaxation. Common B&B techniques compute lower bounds (upper bounds for maximization problems like BIVLP) on the optimal value by solving the continuous relaxation of the integer variables; in this case the sets of variables τ and z. Since the effectiveness of B&B strongly depends on this bound, it is important to analyse how tight the relaxed value is w.r.t. the integral optimal solution.

Flows in constraints (3.7g–3.7j) are bounded to the high value Υ. This means that even for high fractional values of a few variables in τ the capacity of the pipes would be still enough to feed the entire network, e.g.,

$$\tau_{ij}^s = 0.9 \Rightarrow x_{i,\epsilon_{ij}}^s = x_{\epsilon_{ij},i}^s = 0.1\Upsilon$$

In this relation, the valve v_{ij} in the boundary of s can be considered almost closed; nevertheless, the water flowing through that pipe is really high if $\Upsilon >> \delta_{\epsilon_{i,j}}$. Since the aim is to maximize the flow, the continuous relaxation often returns with objective value Υ, even on deep nodes of the search tree. This has a bearing on the general effectiveness of common B&B.

Variable and value selection. At each node of the search tree a common B&B splits the model into two smaller MILP programs (or branches of the tree), following the *divide et impera* paradigm; the aim is to give a integral value to the variables that should be integer in the solution, namely τ and z in this case. To do that, two choices have to be taken on these variables, that are:

- select a variable to branch;
- select which branch is visited first.

Typically the B&B first selects the variable having the closest value to an integer. As far as the visit order, it applies a Depth First Search (DFS) with priority on left side. In Figure 3.11 the generic variable V_b is selected for the branching after the root node r has been solved; the two branches impose $V_b \leq \lfloor v \rfloor$ on the left, and $V_b \geq \lceil v \rceil$ on the right. Then node 1 is solved first. Notice that for binary variables the two branches become $V_b = 0$ on the left, and $V_b = 1$ on the right.

Variables τ and z in (3.9a–3.9d) are binary: the search would explore first branches giving value 0 to those variables. Unfortunately, stating $z_i^s = 0$ is almost meaningless, in fact this choice does not draw a new part of sectorization; anyway, after a choice like this a new node is relaxed and solved, but the new optimal value might be basically the same as before. Stating $\tau_{ij}^s = 0$ means that for sector s the

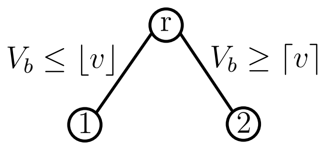


Figure 3.11: A small B&B search tree

pipes (i, ϵ_{ij}) and (ϵ_{ij}, i) are hugely capacitated, but this does not prevent to put a valve there for another sector: this choice is again almost ineffective.

Some MILP solvers can be tuned in order to customize the variable selection policy and branching priority. In this case, it would be better to branch first on the set τ, assigning first the value 1.

Symmetry on sectors' indexes. Consider the solution in Figure 3.10b, sectors S_1 and S_2 can be swapped, and the quality of the solution would be the same. Table 3.2 reports two symmetric solutions for graph in Figure 3.10a; the omitted values are 0 in both solutions. Both "Sol.1" (the solution in Figure 3.10b) and

Table 3.2: Two symmetric solutions for graph in Figure 3.10a

Variables	Sol. 1		Sol. 2	
	S_1	S_2	S_1	S_2
$\tau_{1,2}$	1	0	0	1
$\tau_{2,1}$	1	1	1	1
...		...		
z_1	1	0	0	1
$z_{\epsilon_{1,2}}$	1	0	0	1
z_2	0	1	1	0
$z_{\epsilon_{2,3}}$	0	1	1	0
z_3	0	1	1	0
...		...		
$x_{\sigma,1}$	0	$\delta_{1,2}$	$\delta_{1,2}$	0
$x_{1,\epsilon_{1,2}}$	0	$\delta_{1,2}$	$\delta_{1,2}$	0
$x_{\epsilon_{1,2},P}$	0	$\delta_{1,2}$	$\delta_{1,2}$	0
...		...		
$SD(s)$	0	$\delta_{1,2}$	$\delta_{1,2}$	0
$min_s\{SD(s)\}$	0		0	

"Sol.2" draw the same sectorization and have the same flow values on the network; thus, the objective function is the same. They differ from each other only because of the sector names, that are simply swapped in these two solutions.

Symmetry may strongly worsen the effectiveness of the solving procedure, and it should be addressed to prevent performance deterioration. MILP formulations for GPP are known to be subjected to this effect [97]. Symmetry breaking strategies are discussed further in Section 3.6.

3.4.4 A general bilevel MILP model for the IVLP

As stated in Section 3.1.3 the generic IVLP can be coupled with various objective functions depending on the needs. The most known ones are the maximization of the satisfied demand under some assumptions (e.g., in the worst isolation case, on average, ecc...) and the minimization of the cost of the isolation system. For the BIVLP specialization the objective function in model (3.8a–3.8d) already maximizes the network flow for the worst sector, it can be reformulated then as (3.9a–3.9d) and solved by means of common MILP solvers.

Whenever the objective function does not ensure the maximality of the network flows the BIVLP program (3.9a–3.9d) would not be suitable anymore, being the amount of the satisfied demands of non-maximal sectors incorrect. The IVLP is then generalized as:

$$\min : f(\tau) + g(z) + l(x) \tag{3.10a}$$

$$s.t.$$

$$GP(\tau), \tag{3.10b}$$

$$MFP(\tau^s) \begin{cases} SD(\tau^s) = max : \ x^s_{P,\sigma} \\ \qquad s.t. \\ \qquad\quad FP(\tau^s) \end{cases} \qquad \forall\, s \in S. \tag{3.10c}$$

Since we presume that this model does not ensure in general the maximality of flows and may also want to minimize some flows, it can be seen as a *Bilevel* MILP model. In general the water distribution system of a city is owned and managed by a single commercial entity, and it has no reason to put neither costs nor toll on its own network flows. But for other supplies, like gas, electricity, internet traffic, the provider usually pays for using another infrastructure.

Bilevel optimization [95] provides the framework for modelling optimization problems where two decision makers with conflicting objective functions are involved in a hierarchical relationship. The leader takes its decisions aware of the fact that their value depends on how the follower reacts to such decisions. Here, the leader sets the topology of the isolation system, locating the available valves on the pipes. The follower, sector by sector, maximizes the flow from σ to P on a graph whose topology depends on the boundary valves of the sector; in other words the users maximize the consumption of water compatibly with the pipe capacities and the flow's cost.

Bilevel optimization problems can be tackled by imposing inner problem optimality by adding its optimality conditions, usually expressed as non linear constraints, to the inner problem feasibility constraints, and reformulating the whole problem as a single level optimization problem.

3.4.5 A single level reformulation for IVLP

When the inner problem is a continuous linear programming problem, duality can be exploited to state optimality by way of linear constraints [98]. In our case, given the valves location, each subproblem is a maximum flow whose dual is the minimum capacity cut provided in (3.11a–3.11h).

Every $s \in S$ has its own dual, where there is a dual variable for each constraint of the MFP model, indexed here:

$\omega^s_{i,\epsilon_{ij}}, \omega^s_{\epsilon_{ij},i}, \omega^s_{j,\epsilon_{ij}}, \omega^s_{\epsilon_{ij},j}$, $\forall \epsilon_{ij} \in \overline{\overline{V}}$ are the non negative variables associated to capacity constraints (3.7g–3.7j).

$\omega^s_{\epsilon_{ij},P}$, $\forall\ \epsilon_{ij} \in \overline{\overline{V}}$ are the non negative variables associated to capacity constraints (3.7k).

$\omega^s_{\sigma,i}$, $\forall\ (\sigma, i) \in \overline{\overline{T}}$ are the non negative variables associated to capacity constraints (3.7l).

$\pi^s_i, \forall\ i \in \overline{\overline{V}} \setminus \sigma$ are the unconstrained node potential variables associated to flow balance constraints (3.7c) and (3.7f).

π^s_P and π^s_σ are the potential variables associated to the sink and the source flow balance constraints (3.7d) and (3.7e).

For each sector, an equivalent reformulation of the dual problem of (3.7a-3.7m) is stated below.

$$\begin{aligned}
\min: & \sum_{(i,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Upsilon(1-\tau^s_{ij})(\omega^s_{i,\epsilon_{ij}}+\omega^s_{\epsilon_{ij},i})+ \\
& + \sum_{(j,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Upsilon(1-\tau^s_{ji})(\omega^s_{j,\epsilon_{ij}}+\omega^s_{\epsilon_{ij},j})+ \\
& \qquad + \sum_{(\epsilon_{ij},P)\in\overline{\overline{\Pi}}} (\delta_{\epsilon_{ij}}\omega^s_{\epsilon_{ij},P}) && (3.11a)
\end{aligned}$$

$$\begin{aligned}
\pi^s_P - \pi^s_\sigma &\geq 1, && && (3.11b)\\
\pi^s_i - \pi^s_{\epsilon_{ij}} + \omega^s_{i,\epsilon_{ij}} &\geq 0 && \forall\ (i,\epsilon_{ij})\in\overline{\overline{\Psi}}, && (3.11c)\\
\pi^s_{\epsilon_{ij}} - \pi^s_i + \omega^s_{\epsilon_{ij},i} &\geq 0 && \forall\ (\epsilon_{ij},i)\in\overline{\overline{\Psi}}, && (3.11d)\\
\pi^s_j - \pi^s_{\epsilon_{ij}} + \omega^s_{j,\epsilon_{ij}} &\geq 0 && \forall\ (j,\epsilon_{ij})\in\overline{\overline{\Psi}}, && (3.11e)\\
\pi^s_{\epsilon_{ij}} - \pi^s_j + \omega^s_{\epsilon_{ij},j} &\geq 0 && \forall\ (\epsilon_{ij},j)\in\overline{\overline{\Psi}}, && (3.11f)\\
\pi^s_{\epsilon_{ij}} - \pi^s_P + \omega^s_{\epsilon_{ij},P} &\geq 0 && \forall\ (\epsilon_{ij},P)\in\overline{\overline{\Pi}}, && (3.11g)\\
\pi^s_\sigma - \pi^s_i + \omega^s_{\sigma,i} &\geq 0 && \forall\ (\sigma,i)\in\overline{\overline{T}}, && (3.11h)\\
\omega^s_{\sigma,i}, &\geq 0 && \forall(\sigma,i)\in\overline{\overline{T}}, && (3.11i)\\
\omega^s_{\epsilon_{ij},P} &\geq 0 && \forall(\epsilon_{ij},P)\in\overline{\overline{\Pi}}. && (3.11j)\\
\omega^s_{i,\epsilon_{ij},i}, \omega^s_{\epsilon_{ij},i} &\geq 0 && \forall\ (i,\epsilon_{ij}),(\epsilon_{ij},i)\in\overline{\overline{\Psi}}, && (3.11k)\\
\omega^s_{j,\epsilon_{ij},i}, \omega^s_{\epsilon_{ij},j} &\geq 0 && \forall\ (j,\epsilon_{ij}),(\epsilon_{ij},j)\in\overline{\overline{\Psi}}. && (3.11l)
\end{aligned}$$

The dual objective function coefficients of ω depend on τ. To linearise this non linear expression, for each sector we introduce two non negative variables $\mu^s_{i,\epsilon_{ij}}$ and $\mu^s_{\epsilon_{ij},i}$ for each edge $(i,\epsilon_{ij})\in\overline{\overline{\Psi}}$ and constraints (3.12a-3.12b), which realize the equivalence $\mu^s_{i,\epsilon_{ij}} = \omega^s_{i,\epsilon_{ij}}\tau^s_{ij}$.

$$\begin{aligned}
\mu^s_{i,\epsilon_{ij}} &\leq \omega^s_{i,\epsilon_{ij}} && \forall(i,\epsilon_{ij})\in\overline{\overline{\Psi}}, \forall s\in S, && (3.12a)\\
\mu^s_{i,\epsilon_{ij}} &\leq \Upsilon\tau^s_{ij} && \forall(i,\epsilon_{ij})\in\overline{\overline{\Psi}}, \forall s\in S. && (3.12b)
\end{aligned}$$

$\mu^s_{i,\epsilon_{ij}}$ is no greater than $\omega^s_{i,\epsilon_{ij}}$ and it is forced to 0 when τ^s_{ij} is 0. Since its coefficient in the objective function to be minimized is $-\Upsilon < 0$, then $\mu^s_{i,\epsilon_{ij}}$ will be equal to $\omega^s_{i,\epsilon_{ij}}$ if $\tau^s_{ij} = 1$. The same holds for $(\epsilon_{ij},i),(j,\epsilon_{ij})$, and $(\epsilon_{ij},j)\in\overline{\overline{\Psi}}$.

Now we can replace the objective function in (3.10c) of the inner problem for each $s \in S$ by constraint (3.13a–3.13b) that imposes the well known *max flow – min cut* optimality condition.

$$\begin{aligned}
\gamma^s = &\sum_{(i,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Big(\Upsilon(\omega^s_{i,\epsilon_{ij}} + \omega^s_{\epsilon_{ij},i}) - \Upsilon(\mu^s_{i,\epsilon_{ij}} + \mu^s_{\epsilon_{ij},i})\Big) \\
&\sum_{(j,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Big(\Upsilon(\omega^s_{j,\epsilon_{ij}} + \omega^s_{\epsilon_{ij},j}) - \Upsilon(\mu^s_{j,\epsilon_{ij}} + \mu^s_{\epsilon_{ij},j})\Big) \\
&+ \sum_{(\epsilon_{ij},P)\in D} (\delta_{\epsilon_{ij}} \omega^s_{\epsilon_{ij},P}), \qquad (3.13a)\\
\gamma^s = &\, x^s_{P,\sigma}. \qquad (3.13b)
\end{aligned}$$

Finally, let us call $CP(\tau^s)$ the linear constraints of the minimum cut program in (3.11b–3.11l,3.13a), the single level MILP for the general IVLP is given by:

$$\begin{aligned}
&min : f(\tau) + g(z) + l(x) && (3.14a)\\
&s.t. && \\
&GP(\tau), && (3.14b)\\
&MFP(\tau^s)\begin{cases} \gamma^s = x^s_{P,\sigma} \\ FP(\tau^s) \\ CP(\tau^s) \end{cases} \qquad \forall\, s \in S. && (3.14c)
\end{aligned}$$

Depending on the real objective function this model is now solvable by common MILP solvers; it is even suitable to solve the BIVLP, but there is no reason to prefer this program to the other in (3.9a–3.9d), which involves a number of variables and constraints by far lower.

3.5 A Logic Programming formulation

Logic Programming (LP) [26] is a declarative paradigm in Artificial Intelligence (AI) [99] based on the *first order* logic, and the most famous LP language is Prolog [100]. A program in Prolog is typically queried by means of an inference algorithm such as Selective Linear Definite clause resolution (SLD), well known implementations are: BProlog [101], ECL^{i}PS^e [102], SICStus Prolog [103], SWI-Prolog [104]. Another logic language is called Answer Set Programming (ASP) [28, 105, 106] and some implementations are: Cmodels [107], DLV [108], Clingo [109].

COPs often are addressed in LP by Constraint Logic Programming (CLP) and ASP. In particular a Constraint Logic Programming on Finite Domain (CLP(FD)) program has been already proposed for the BIVLP [34]. In the last decade ASP has come up as a competitive technology to address COPs, both for its high declarative capability and the effectiveness of the off-the-shelf ASP solvers.

In the next pages we go further on the ASP syntax, afterwards three ASP encodings are proposed for the BILVP. The encodings in Section 3.5.2 have been proposed in [80], the latter encoding described in Section 3.5.3 has been proposed in [81].

3.5.1 Answer Set Programming

Answer Set Programming (ASP) is a class of logic programming languages that rely on the stable model semantics [110], also known as answer set semantics. We assume a basic familiarity with logic programming and its syntax; for an introduction one can refer to [26].

First of all, all strings starting with an upper case letter are considered to be variables (e.g., `Var`); all the others, starting with a lower case latter are constants (e.g., `val`). An *atom* is an expression $\mathtt{p(t_1,\ldots,t_n)}$, which is a *predicate* of arity n consisting of a *functor* p, and n *terms* t. Two atoms refer to the same predicate if they have same functor and same arity. For brevity, all predicates p of arity n are often referred to as `p/n`. A constant is also a predicate with $n = 0$. An atom $\mathtt{p(t_1,\ldots,t_n)}$ and its negation $\neg\mathtt{p(t_1,\ldots,t_n)}$ are referred to as *literals*. A logic program consists of a set of rules

$$l_1 \vee \cdots \vee l_k \leftarrow l_{k+1}, \ldots, l_m, \text{not}\, l_{m+1}, \ldots, \text{not}\, l_n$$

where l_i's from l_1 to l_k are literals of the *head* of the rule, from l_{k+1} to l_m are positive literals of the *body*, and from l_{m+1} to l_n are negative literals of the *body*; the head is this general form a disjunctive clause, while the body is a conjunction. Rules with single literal in the head and no negative literals in the body, like

$$l_1 \leftarrow l_2, \ldots, l_m$$

are called *nlp* rules (from *normal logic program*). Rules with empty body are also called *facts* while rules with empty head are called *Integrity Constraint* (IC):

$$\leftarrow l_2, \ldots, l_m, \ldots, \text{not}\, l_{m+1}, \ldots, \text{not}\, l_n$$

and their body must evaluate to *false* in any model of the program. In LP programs symbol $\leftarrow$ is substituted by `:-` and rules always end with a dot:

$$l_1 \text{ :- } l_2, \ldots, l_m.$$

Literals and rules containing no variables are called *ground*. We denote as $gr(r)$ all possible instantiations of the rule r of the program Π, on the basis of ground facts of the program. The *ground instantiation* of Π is a program consisting of all ground instances of rules in Π, i.e., $gr(\Pi) = \bigcup_{r\in\Pi} gr(r)$. For any set M of atoms from Π, let Π_M be the program obtained from Π by deleting (i) each rule that has a negative literal $\neg B$ in its body with $B \in M$ and (ii) all negative literals in the bodies of the remaining rules. Since Π_M is negation free, it has a unique minimal Herbrand model. If this model coincides with M, then M is a Stable Model of Π [110]. In general, a program may have zero, one, or more answer sets.

Consider the following logic program Π_0 (and its grounding $gr(\Pi_0)$):

$$\Pi_0 \left\{ \begin{array}{l} \texttt{p(1).} \\ \texttt{p(2).} \\ \texttt{q(X,Y):- p(X), p(Y),} \\ \quad\;\texttt{X!=Y, not q(Y,X).} \end{array} \right. \qquad gr(\Pi_0) \left\{ \begin{array}{l} \texttt{p(1).} \\ \texttt{p(2).} \\ \texttt{q(2,1):- not q(1,2).} \\ \texttt{q(1,2):- not q(2,1).} \end{array} \right.$$

The program Π_0 has two answer sets:

$$S_1 = \{p(1).\ p(2).\ q(1,2).\} \text{ and } S_2 = \{p(1).\ p(2).\ q(2,1).\}$$

Among the various dialects and implementations of ASP, we use the language of the grounder Gringo [111], that extends the basic logic programming syntax with a number of features. We show the most relevant to this article, while the interested reader can refer to [111] for a complete description.

Intervals If an atom $a(i..j)$, where i and j are integers, is in a clause, the clause is repeated for all possible values k such that $i \leq k \leq j$. For example, the fact $a(1..4)$. is expanded to four facts, namely `a(1). a(2). a(3). a(4).`

Conditions allow for instantiating variables to collections of terms within a single rule. A condition has the syntax $a(X) : c(X)$, and it is expanded to as many $a(X)$ atoms as those making true the atom $c(X)$. For example, consider the program Π_1 and its grounding $gr(\Pi_1)$

$$\Pi_1 \left\{ \begin{array}{l} \texttt{p(1..3).} \\ \texttt{q:- r(X):p(X).} \end{array} \right. \qquad gr(\Pi_1) \left\{ \begin{array}{l} \texttt{p(1).\ \ p(2).\ \ p(3).} \\ \texttt{q:- r(1), r(2), r(3).} \end{array} \right.$$

Notice that the condition in the body is expanded to a conjunction.

Counting [112] If $a_1, a_2, a_3, \ldots$ are atoms, and l and u are integers, the aggregate

$$l\ \{a_1, a_2, a_3, \ldots\}\ u$$

is true for every set S of atoms including from l to u members of $\{a_1, a_2, a_3, \ldots\}$, i.e., $l \leq |\{a_i \in S\}| \leq u$. Trivial bounds can be omitted.

Summation If $a_1, a_2, a_3, \ldots$ are atoms and $v_1, v_2, v_3, \ldots$ are integers, the aggregate

$$l\ \sharp\mathbf{sum}[a_1 = v_1, a_2 = v_2, a_3 = v_3, \ldots]\ u$$

is true for every set S of atoms such that the sum of v_i over included members a_i of $\{a_1, a_2, a_3, \ldots\}$ is in the interval $[l, u]$:

$$l \leq \sum_{i:a_i \in S} v_i \leq u$$

Choice rules and optimisation statements In combinatorial optimisation the answer sets of an ASP program represent the feasible solutions of the problem. To compute the optimal answer set, an ASP solver needs to guess on the decision variables of the problem and optimise an aggregated value that depends on those variable. A choice is made on a set of candidates and it is stated by means of a counting aggregate, as following:

$$l\ \{c_1, c_2, c_3, \ldots\}\ u.$$

This statement asks to the solver to look for answer sets containing a number of facts c ranging between l and u.

On some predicates of the program a cost function can be maximized (or minimized) by an optimisation statement, in which a cost is assigned to the variables of interest. For example, the first statement below maximizes the sum of the c's costs, whereas the second minimizes it:

$$\sharp\mathbf{maximize}[p_1 = 1, p_2 = 2, p_3 = 3, \ldots].$$

$$\sharp\mathbf{minimize}[p_1 = 1, p_2 = 2, p_3 = 3, \ldots].$$

The following ASP program:

```
1 {a,b,c,d} 2.
b :- c.
#maximize[a=1,b=2,c=3,d=4].
```

is maximized by the answer set {`b. d.`}. A possible translation of this ASP program into the mathematical analogue would be:

$$max : 1a + 2b + 3c + 4d \tag{3.15a}$$
$$c \leq b \tag{3.15b}$$
$$a + b + c + d \geq 1 \tag{3.15c}$$
$$a + b + c + d \leq 2 \tag{3.15d}$$
$$a, b, c, d \in \{0, 1\} \tag{3.15e}$$

Rule 1 can be translated into the three mathematical relations in Eq. (3.15c–3.15e); Rule 2 is Eq. (3.15b); finally Rule 3 is the optimisation statement (3.15a). However this is a particular case, in fact ASP programs can not always be translated into MILP by following this schema.

ASP solvers

Usually, ASP solvers [107, 108, 112–114] work in two stages. In the first, called *grounding*, the program is converted into an equivalent ground program. The ground program can be very large, depending on the input program. The second stage is devoted to looking for stable models (answer sets) of the ground program.

ASP Formulations for the BIVLP

We present three different ASP programs to the Bottleneck Isolation Valves Location Problem. Two of them share a common set of rules, which define the reachability of each pipe from sources varying the isolated pipe, so they are presented together in Section 3.5.2. These two approaches differ in the definition of the worst isolation case, either in terms of a single pipe or a group of pipes.

In the third approach, described in Section 3.5.3, reachability from tanks is defined only for the sectors, and the worst isolation case is computed by varying the isolated sector.

In all cases, the input data consists of a set of facts that describe the graph of the water distribution network. Nodes are given as facts of the form `node(X)`, while labelled edges are facts of the form `e(I,J,D)`, where `I` and `J` are nodes of the graph, and `D` is the demand associated to the edge. The instances of `e(I,J,D)` are supposed to be symmetric. The sources of water are usually tanks, and they are given in the form `tank(N)`, where `N` is the name of the node. The maximum number of available valves N_v is given in input as a fact `nv(`N_v`)`. E.g., the network in Figure 3.1 is represented as:

```
tank(1). node(1..8). nv(7).
e(1,2,4). e(1,4,12). e(2,3,7). e(2,5,5). e(3,6,1).
e(4,5,9). e(5,6,2). e(5,7,5). e(6,8,2). e(7,8,6).
```

To simplify notation, we will also use the following definitions:

```
1 e(X,Y):- e(X,Y,_).
2 sym(X,Y):- e(X,Y).
3 sym(Y,X):- e(X,Y).
4 swap(e(X,Y),e(X,Y)):- e(X,Y).
5 swap(e(X,Y),e(Y,X)):- e(X,Y).
6 adj(COM,U1,U2,e(X,Y),e(W,Z)):-
       not tank(COM),
       swap(e(X,Y),e(COM,U1)),
       swap(e(W,Z),e(COM,U2)), U1!=U2.
```

Predicate `sym/2` defines the symmetric edge of the graph. Predicate `swap/5` contains the edge $e(X, Y)$ (Rule 4) and its symmetric (Rule 5). Predicate `adj/5` defines pairs of adjacent edges: two different pipes $e(X, Y)$ and $e(W, Z)$ are adjacent if they share exactly one junction node, namely `COM`. The terms `U1` and `U2` are the uncommon nodes the adjacent have. Since the adjacency is exploited later to model the reachability of pipe from sources, the edges having the tank as common node are not considered to be adjacent; this condition is stated in Rule 6 by the negation "`not tank(COM)`".

3.5.2 ASP Programs based on Pipe Reachability

As mentioned in the problem description (Section 3.1.1), each edge should be isolable by closing some set of valves.

We can isolate network patches by closing valves. However, the set of closed valves depends on where the damaged pipe is; for this reason, we define a predicate `close/2`. The meaning is that `close(v(A,B),e(X,Y))` is true iff the valve `v(A,B)` (located on the edge `e(A,B)` close to node `A`) will be closed when the pipe `e(X,Y)` is broken. Predicate `close/2` is defined in Code 3.1. Rule 1 states that, for every (tentatively isolated) edge `e(X,Y)`, at most N_v valves can be closed. Rule 2 states that if a valve is closed (for at least one broken pipe), then there must be a valve in that position. Predicate `valve/2` is the intended answer, as it represents the positioning of the valves in the isolation system; the number of valves should be exactly N_v (rule 3).

Code 3.1: ASP encoding for isolation of a pipe based on *Pipe Reachability*

```
1 1 { close(v(A,B),e(X,Y)): sym(A,B) } N_v:-
    e(X,Y), nv(N_v).
2 valve(A,B):- close(v(A,B),_).
3 :- nv(N_v), not N_v { valve(A,B): symm_e(A,B) } N_v.
  %reachability of pipes
4 reached(e(A,B),e(X,Y)):- e(X,Y),
          swap(e(A,B),e(T,D)), tank(T),
          not close(v(T,D),e(X,Y)).
5 reached(e(A,B),e(X,Y)):- e(X,Y),
          adj(COM,U1,U2,e(A,B),e(C,D)),
          not close(v(COM,U1),e(X,Y)),
          not close(v(COM,U2),e(X,Y)),
          reached(e(C,D),e(X,Y)).
6 :- reached(e(X,Y),e(X,Y)).
```

Rules 1, 2 and 3 assure that, for each (damaged) pipe, there is a subset of the installed valves that will be closed, but there is no knowledge of which users (or, which pipes) will be reached by the water in each scenario.

For this purpose, predicate `reached(e(A,B),e(X,Y))` explains which edges `e(A,B)` are reachable by the sources when pipe `e(X,Y)` is damaged. In particular (rule 4), pipe $e_{A,B}$ is reached if one of its endpoints is a tank and near the tank there is no valve, or if there is a valve but it is not closed when the damaged edge is $e_{X,Y}$ (Figure 3.12). Otherwise (rule 5), $e_{A,B}$ is reached if at least one of its adjacent edges ($e_{C,D}$ in Figure 3.13) is reached and no valves are closed between the two (again, when the damage is in $e_{X,Y}$). Finally, an integrity constraint (Rule 6) imposes that a broken pipe should not be reachable by water.

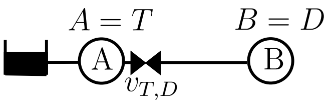


Figure 3.12: Example network

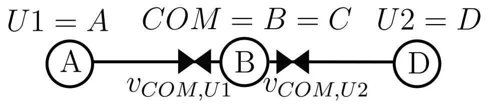


Figure 3.13: Example network

We propose two different ways to extend the above program in order to define and minimize the undelivered demand in the worst-case isolation: the "*Pipe Isolation*" encoding, and the "*Extended Sectors*" encoding.

Pipe Isolation Encoding

As explained earlier, the objective is to maximize the satisfied demand (or, equivalently, minimize the unsatisfied demand) in the worst case. As a first attempt, we might use the ♯sum aggregator operator [115], and define the satisfied water demand when some pipe $e_{X,Y}$ is broken ($SD(e_{X,Y})$) as the sum of the demands of the pipes reached by water:

```
sd(SD,e(X,Y)):- e(X,Y),
     SD = #sum[ reached(e(A,B),e(X,Y))=D
                    : e(A,B,D) ].
```

Unfortunately, this type of aggregation leads to an explosion of the ground program, especially if the single demands `D` can take large integer values. Another way to find the minimum (total) satisfied demand is by means of pairwise comparisons amongst `reached/2` atoms, varying the "broken" pipe, using the aggregator ♯`sum` as a conditional operator. Predicate `lower(e(X,Y),e(W,Z))`, defined by the rule 7 in Code 3.2, is true if the satisfied demand in case the edge $e_{X,Y}$ is broken is lower than the satisfied demand when the broken edge is $e_{W,Z}$. In mathematical notation, clause 7 could be written as

$$e_{X,Y} \preceq e_{W,Z} \leftarrow \sum_{\substack{reached(e_{A,B},e_{X,Y})\\ D_n=w(e_{A,B})}} D_n + \sum_{\substack{reached(e_{C,D},e_{W,Z})\\ D_m=w(e_{C,D})}} -D_m \leq 0$$

In this way, the ♯`sum` aggregator sums with a positive sign the satisfied demand when the broken edge is the first ($e_{X,Y}$), and with a negative sign the satisfied demand when the broken edge is the second ($e_{W,Z}$). The body evaluates to *true* if this algebraic sum is negative or null. The grounder instantiates rule `lower/2` for each pair of edges, so that we move to the solver the task to check whether the sum is indeed less than or equal to 0 and it will return answer sets containing only the actual comparison results.

Now we compute the worst case: it occurs when damaging the edge $e_{X,Y}$ whose isolation provides a satisfied demand less than (or equal to) the satisfied demand due to the isolation of any other edge:

```
worst(e(X,Y)):- e(X,Y),
      lower(e(X,Y),e(W,Z)): e(W,Z).
```

We now know the pipe that gives minimum satisfied demand; our objective is to find the best valve positioning such that the satisfied demand, in the worst case, is maximum (i.e., $max : min_{e_{X,Y}}\{SD(e_{X,Y})\}$). Since we need to compute the value of the corresponding satisfied demand and maximize it, one is tempted to use the following formulation:

```
min_sd(e(A,B),D):- e(A,B,D),
        reached(e(A,B),WorstCase),
        worst(WorstCase).
#maximize[ min_sd(e(A,B),D)=D ].
```

However, the minimum is not unique, as there can be two or more sectors that, when isolated, provide the same delivered demand. Also, the sector corresponding to minimal satisfied demand may contain more than one pipe: if one of those pipes is broken, they will all provide the same delivered demand. When the minimum is not unique, with the above definitions we would sum one contribution for each of the equally good minima. In order to compute the correct delivered demand, we select one of the minima, through a lexicographic comparison, with the rules 9 and 10 of Code 3.2.

Code 3.2: The *Pipe Isolation Encoding*

```
7  lower(e(X,Y),e(W,Z)):- e(X,Y), e(W,Z),
     #sum[ reached(e(A,B),e(X,Y))=Dn: e(A,B,Dn),
           reached(e(C,D),e(W,Z))=-Dm: e(C,D,Dm) ]0.
8  lower_lexico(e(X,Y),e(X,Y)):- lower(e(X,Y),e(X,Y)).
9  lower_lexico(e(X,Y),e(W,Z)):- % e_X,Y ≺ e_W,Z
     lower(e(X,Y),e(W,Z)), not lower(e(W,Z),e(X,Y)).
10 lower_lexico(e(X,Y),e(W,Z)):- % same delivered demand
     lower(e(X,Y),e(W,Z)), lower(e(W,Z),e(X,Y)),
     (X,Y)<(W,Z). % this implements: X<W ∨ (X==W ∧ Y<Z)
11 min_sd(e(A,B),SD):- e(A,B,SD), e(X,Y),
     lower_lexico(e(X,Y),e(W,Z)): e(W,Z),
     reached(e(A,B),e(X,Y)).
12 #maximize[ min_sd(e(A,B),SD)=SD: e(A,B) ].
```

Finally, we maximise the sum of the delivered demands in the worst-case, those defined by rule 11.

This is maybe the most declarative encoding for the BILVP. Notice that the Pipe Isolation encoding does not need to be dimensioned on N_s. This may also be a good point with respect to the solving algorithm, because no symmetries on sectors' name are here introduced.

However, the choice strategy is not very clever; in fact, the solver looks how to close some valves in order to isolate each pipe and tries to maximize the satisfied demand; but only the configurations of closed valves for which the intersection among the whole set of pipe consists of N_v installed valves are feasible and can be candidate for the optimal solution. A better choice would be instead to guess first the general positioning of the available valves, then to compute in a deterministic fashion the set of closed valves for each pipe.

Extended Sectors Encoding

The previous Pipe Isolation encoding does not explicitly define the concept of *sector*, but it maximizes the satisfied water demand of the worst-case isolation. In Code 3.3, we associate to each pipe a sector identifier, represented as an integer.

In the hydraulic literature, a sector is simply the sub-graph encircled by a set of valves, so each pipe belongs to exactly one sector; here we call *extended sector* the set of unreachable pipes given an isolation. As explained in the introduction, isolating a sector S can isolate a part of the network larger than the sector S itself, due to the effect of unintended isolation. In such a case, the extended sectors do not represent a partition of the network: some pipes can belong to two (or more) extended sectors. E.g., for the network in Figure 3.1, edge $e_{7,8}$ belongs of course to sector $\{e_{6,8}, e_{7,8}\}$, as well as to the extended sector $\{e_{1,2}, e_{2,5}, e_{5,6}, e_{5,7}, e_{2,3}, e_{3,6}, e_{6,8}, e_{7,8}\}$, that is the set of edges that will be de-watered in case of damage of, e.g., edge $e_{1,2}$.

Code 3.3 reports rules aimed to the definition of the worst extended sector, and to the minimization of the related undelivered demands. Predicate `s/1` (rule 13) declares the possible sector identifiers (assuming there is some maximum number N_s of extended sectors). Predicate `ext_sector(e(A,B),S)` says that the edge `e(A,B)` belongs to the extended sector `S`: rule 14 states that each edge belongs to at least one extended sector, while rule 15 states that two edges belong to the same extended sector if, whenever one is isolated, the other one is unreachable as well. In this case, if $e_{A,B}$ is not reachable due to an indirect side effect of the isolation of $e_{C,D}$, then $e_{A,B}$ belongs to two or more different extended sectors, at least one of which is in common with $e_{C,D}$.

Code 3.3: The *Extended Sector Encoding*

```
13 s(1..N_s).
14 1 { ext_sector(e(A,B),S): s(S) } :- e(A,B).
15 ext_sector(e(A,B),S):- e(A,B), ext_sector(e(C,D),S),
     not reached(e(A,B),e(C,D)).
16 empty(S):- s(S), not ext_sector(e(A,B),S): e(A,B).
17 greater(S1,S2):- s(S1), s(S2), S1!=S2, not empty(S1),
     0 #sum[ext_sector(e(A,B),S1)=D1: e(A,B,D1),
            ext_sector(e(C,D),S2)=-D2: e(C,D,D2)].
18 maxSect(MaxS):- s(MaxS), not empty(MaxS),
     greater(MaxS,S): MaxS!=S: s(S).
```

```
19 uniqueMax(S):- S=#max[ maxSect(S1)=S1 ].
20 max_ud(e(A,B),UD):-
     ext_sector(e(A,B),S), uniqueMax(S), e(A,B,UD).
21 #minimize [ max_ud(e(A,B),UD)=UD ].
```

In order to find the sector that determines the maximum service disruption if isolated, we proceed with pairwise comparisons among all sectors, similarly to the Pipe Isolation encoding. Rule 17 defines an order relation, and states that sector `S1`$\succ$`S2` if `S1` is not empty and the sum of the demands of its edges is greater than the sum of weights of `S2`. Then, the worst disservice is determined by the sector `MaxS` such that ($\forall$`S`) `MaxS`$\succeq$`S` (rule 18).

As in the Pipe Isolation encoding, two or more extended sectors may determine the same disservice, so we arbitrarily select one by choosing (among the equally worst-case ones) the one with the maximum identifier (rule 19). Finally, we minimize the sum of the demands of the pipes belonging to the sector of the unique worst-case, defined by rule 20, that are the undelivered demand of the worst isolation case (i.e., $min : max_{s \in S}\{UD(s)\}$).

This encoding chooses to which sector a pipe must belong and it compares each pair of sectors in order to identify the worst one. This is better than the Pipe Isolation encoding, because the number of comparisons is lower (in fact the number of sectors is typically lower than the number of pipes, see Section 3.6.1). However the solver has to guess also the binding between pipes and sectors and this yields symmetries and the exploration of an amount of infeasible choices.

3.5.3 An ASP Program based on Sector Reachability

The third encoding does not check the reachability of each single pipe from the tank, but checks only reachability of non-empty *sectors* among the N_s available. The logic program listed in Code 3.4 models sectors as clusters of *adjacent* pipes bounded by valves. Predicate `valve/2` defines the valve placement, with N_v valves (rule 22).

Code 3.4: ASP encoding based on *Sector Reachability*

```
22 N_v { valve(A,B): sym(A,B) } N_v.
23 s(1..N_s).
24 1 { sector(e(A,B),S): s(S) } 1:- e(A,B).
25 :- adj(COM,U1,U2,e(A,B),e(C,D)), s(S), sector(e(A,B),S),
      not sector(e(C,D),S),
      0 { valve(COM,U1),valve(COM,U2) } 0.
26 boundary(e(A,B),e(C,D),S1,S2):- adj(_,_,_,e(A,B),e(C,D)),
           sector(e(A,B),S1),
           sector(e(C,D),S2), S1!=S2.
27 root(S) :- 1 {tank(A),tank(B)} 1, sector(e(A,B),S).
28 reached(Sr,I):- root(Sr), s(Sr), s(I), Sr!=I.
29 reached(S1,I):- s(I), s(S1), s(S2), S1!=S2, I!=S1, I!=S2,
          not empty(I), not empty(S1),
          not empty(S2), reached(S2,I),
          1 { boundary(e(A,B),e(C,D),S1,S2),
              boundary(e(A,B),e(C,D),S2,S1) }.
30 ext_sector(e(A,B),S):- sector(e(A,B),S).
31 ext_sector(e(A,B),S):- sector(e(A,B),S1), s(S), S!=S1,
             not sector(e(A,B),S),
             not empty(S), not reached(S1,S).
```

We associate an integer identifier to each sector (rule 23) and state that each pipe belongs exactly to one sector (rule 24). The integrity constraint 25 imposes that if two adjacent pipes belong to different sectors, there must be at least one valve in between. Rule 26 defines the concept of *boundary* between sectors: two adjacent pipes `e(A,B)` and `e(C,D)` are in the boundary between sectors `S1` and `S2` if `e(A,B)` belongs to `S1` and `e(C,D)` belongs to `S2`.

To properly handle unintended isolation, we define for each sector isolation case the reachability of the other sectors from tanks. We notice that sectors directly connected to the tank cannot be dewatered due to unintended isolation; then, we define them as *root* sectors (rule 27), and state they are always reachable when a different sector `I` is isolated (rule 28). Instead, the reachability of each other sector depends on the reachability of its adjacent sectors when sector `I` is isolated: sector `S1` is reachable if at least one of its adjacent sectors (`S2`) is reached. Otherwise, if `S1` is not reached when `S` is isolated, `S1` is an unintended isolation of `S`, and its pipes belong to the same extended sector `S` (rule 31).

As in the two previous encodings, in order to define the worst case isolation, in Code 3.5 we compute all the possible pair-wise comparisons between extended sectors; then we define the worst extended sector as the one which determines the greatest unsatisfied demand, and we break ties using the sector identifier.

Code 3.5: Computing $min : max_{s \in S}\{UD(s)\}$

```
33 :- s(S1), s(S2), S1<S2, empty(S1), not empty(S2).
34 greater(S1,S2) :- s(S1), s(S2), S1<S2,
          not empty(S1), empty(S2).
35 greater(S1,S2) :- s(S1), s(S2), S1<S2,
          not empty(S1), not empty(S2),
          Isolated=S1, not reach(S2,Isolated).
```

```
36 greater(S1,S2) :- s(S1), s(S2), S1 < S2,
           not empty(S1), not empty(S2),
           Isolated=S1, reach(S2,Isolated),
           0 [ ext_sector(e(X,Y),S1)=D1: e(X,Y,D1),
               ext_sector(e(A,B),S2)=-D2: e(A,B,D2)].
37 greater(S1,S2) :- s(S1), s(S2), S1>S2,
           not empty(S1), not empty(S2),
           not greater(S2,S1).

39 :- s(S1), s(S2), S1!=S2,
       not empty(S1), not empty(S2),
       Isolated=S2, not reach(S1,Isolated),
       greater(S1,S2).

41 worst(S):- s(S), not empty(S),
         greater(S,S2): s(S2): S2!=S.

43 max_ud(e(X,Y),UD) :- ext_sector(e(X,Y),S),
          worst(S), e(X,Y,UD).

45 #minimize [ max_ud(e(X,Y),UD)=UD: e(X,Y) ].
```

Notice that in this encoding we do not use the `closed_valve` predicate, but we check reachability caused by *sectors* disconnections, instead of *pipes* disconnections. This approach (called the *Sectors*-based encoding) is rather different from the previous two and strongly reduces the search space.

However, as well as the extended sectors encoding based on pipe reachability, also this encoding guesses the binding between pipes and sectors (Rule 24); this introduces symmetries and may lead to an amount of infeasible choices during the search.

Next section describes thoroughly how to dimension the number of sectors and describes several symmetry breaking strategies, in order to reduce the search space. Such strategies are applied both for the MILP approach in Section 3.4 and for the ASP approach proposed just above.

3.6 Search space reduction strategies

Both the MILP approach (Section 3.4) and the ASP approach (Section 3.5) are affected by the following weaknesses:

- the dimension of the programs depends on the number of sectors N_s;

- symmetry on the sectors names;
- symmetry on valve positioning;
- redundancy of particular valves configurations.

Any point of these may likely influence the solving algorithm and possibly worsen the effectiveness.

Following the order above, this section describes in details each weakness and proposes some strategies to face it down; whenever possible a practical formulation of the strategies for both the approaches will be provided either.

3.6.1 Estimation of the Number of Sectors

Clearly, the value N_s should not be too small, because we could wrongly rule out some solutions: those in which the number of sectors is actually higher than N_s. On the other hand, the search space explored by the MILP and ASP solvers depends on the N_s parameter: the ASP Rules 14 and 24 associate each edge with a sector identifier, so the size of the search space is proportional to $|E|^{N_s}$; similarly, the MILP model assigns edges and nodes to the sectors, and the search space grows with $(|E| + |V|)^{N_s}$.

The computation time depends significantly on the maximum number of sectors N_s, so for optimal performance it is important to have a value as tight as possible. The following lemma will be useful to compute a bound, based on the number of valves.

Lemma 2. *Given a connected graph and a placement of N valves A_N that produced S sectors, if we add another valve the number of sectors cannot be more than $S + 1$.*

To prove the lemma, we rely on the graph $\overline{G}$ described in Section 3.4.1. With this definition, each edge $(i, j) \in \overline{\Psi}$ can host up to one valve, so defining a partition through valves amounts to defining a set of edges to be removed from $\overline{G}$.

Proof 2. Adding one valve amounts to removing one edge in $\overline{G}$. Removing an edge (i, j) can have two possible effects:

1. if there is no other path from i to j, it disconnects the endpoints i and j, that are now in different sectors. We obtain $S+1$ sectors with $N+1$ valves.
2. if there exists another path from i to j, then the number of sectors is unchanged, and we have S sectors with $N + 1$ valves.

□

We are now ready to prove the following Theorem:

Theorem 3. *Given a connected graph, the number of sectors that can be obtained by adding N_v valves is at most N_v. This bound is strict.*

Proof. Given any valve placement, this placement can be obtained by a procedure that adds the valves in sequence, one after the other. We start with a graph with only one valve: at least one valve is necessary to disconnect the network from the tank. By induction, suppose that in the described procedure we have added N valves, obtaining S sectors; by the inductive hypothesis, $S \leq N$.

By Lemma 2, by adding a valve, the number of sectors increases at most by 1. This means that the number of sectors N_s is bounded by the number of valves: $N_s \leq N_v$. This bound is strict, and the case $N_s = N_v$ happens, e.g., if the network graph is actually a tree. □

Theorem 3 provides an upper bound on the number of sectors valid in *any* connected graph. A tighter upper bound, depending also on network topology, can be computed in polynomial time as

$$UB_{\sharp S}(N_v) = cc + N_v - N_v^{min} \tag{3.16}$$

where N_v^{min} is the minimum number of valves that disconnect the whole network from tanks (that is the sum of the degrees of tank nodes) and *cc* is the number of connected components obtained after tank nodes have been removed.

However, since the search space depends exponentially on the N_s parameter, it could be convenient to find out the *actual* maximum number of sectors N_s^* that can be obtained on the *actual* graph given at most N_v valves. This could provide a better value to be fed (as N_s parameter) to the ASP/MILP program that computes the valve positioning, with the schema shown in Fig. 3.14 with solid lines.

Minimizing the number of sectors

Both the MILP and ASP programs can be adapted in a straightforward way in order to compute the tightest bound on N_s, called N_s^*. We call this program $Max(\sharp S)$. This is actually an optimisation problem by default, but it is clearly easier to solve than the BIVLP; moreover, under certain assumptions it can be solved as a satisfiability problem, as described below. The idea is to design a two–step solving procedure; the first step solves $Max(\sharp S)$, the new bound N_s^* is then the input of the BIVLP solver, as shown in Figure 3.14. Since the number of

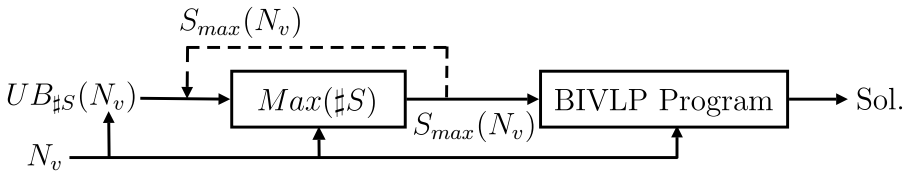


Figure 3.14: The two-step solving procedure

variables and constraints of the second step grows with N_s, the whole computing time of this architecture could be lower than the BIVLP using $UB_{\sharp S}(N_v)$. Further details about the complexity of $Max(\sharp S)$ are given in the next paragraph. Experimental results are discussed in Section 3.7.

The maximizing program $Max(\sharp S)$ is in turn dimensioned on the N_s parameter; in general this can be provided by Theorem 3. However, in many cases the objective is finding the Pareto front [7], and this can be achieved by running a sequence of optimisations, with increasing number of valves [34]. If this is the case, one can use the following Corollary to provide a tighter bound:

Corollary 4. *Given a connected graph, if with N_v valves the maximum number of sectors is $S_{max}(N_v) = N_s^*$, then with $N_v + 1$ valves, the maximum number of sectors cannot be more than $S_{max}(N_v + 1) = N_s^* + 1$.*

Proof. Let A_{N_v} be a valve positioning with N_v valves. Since $S_{max}(N_v)$ is the maximum number of sectors obtainable with N_v valves, the number of sectors $S(A_{N_v})$ corresponding to assignment A_{N_v} is $S(A_{N_v}) \leq S_{max}(N_v)$.

Let us now add another valve to A_{N_v}, obtaining the assignment A_{N_v+1}. By Lemma 2, the new valve positioning A_{N_v+1} has at most $S(A_{N_v+1}) \leq S(A_{N_v})+1 \leq N_s^* + 1$ sectors.

Since $S(A_{N_v+1}) \leq N_s + 1$ holds for any valve positioning A_{N_v+1}, then

$$\max_{A_{N_v+1}} S(A_{N_v+1}) \leq N_s + 1.$$

$\square$

To sum up, the input parameter to dimension the number of available sectors into the general $Max(\sharp S)$ program is:

$$S_{dim}(N_v) = \begin{cases} UB_{\sharp S}(N_v) & \text{if } S_{max}(N_v-1) \text{ unknown} \\ S_{max}(N_v-1)+1 & \text{if } S_{max}(N_v-1) = S_{max}(N_v-2)+1 \\ S_{max}(N_v-2)+1 & \text{if } S_{max}(N_v-1) = S_{max}(N_v-2) \end{cases} \tag{3.17}$$

Moreover, since

$$S_{max}(N_v - 1) \leq S_{max}(N_v) \leq S_{max}(N_v - 1) + 1$$

the maximum number of sector is:

$$S_{max}(N_v) = S_{max}(N_v - 1) + I \qquad I \in \{0, 1\} \tag{3.18}$$

Knowing $S_{max}(N_v - 1)$, the optimisation of $Max(\sharp S)$ for N_v is equivalent to test in Eq. (3.18) whether $I = 1$ or not, that is a satisfiability problem. Section 3.7 discusses experimental results about $Max(\sharp S)$.

Implementing $Max(\sharp S)$ **in MILP.** The MILP program in $GP(\tau) \equiv$ (3.2a-3.2c,3.3a-3.3d) can be extended in the following way:

$$\max : \sum_{s \in S} y_s \tag{3.19a}$$

$$GP(\tau) \tag{3.19b}$$

$$\sum_{i \in \overline{V}} z_i^s \geq |\overline{V}| y_s \qquad \forall\, s \in S \tag{3.19c}$$

$$y_s \in \{0, 1\} \qquad \forall\, s \in S \tag{3.19d}$$

To count the *non-empty sectors* the variables y_s are introduced. Constraints (3.19c) imposes that every sector contains something. The number of non-empty sectors is then maximized (Eq. 3.19a).

Finally, to implement the satisfiability version and to test whether $S_{max}(N_v) = S_{max}(N_v - 1) + 1$, Eq. (3.19a) can be omitted and Eq (3.19a) becomes

$$\sum_{i \in \overline{V}} z_i^s > 0.$$

This program has a solution only if a valve positioning with N_v valves can yield $S_{max}(N_v - 1) + 1$ sectors.

Implementing $Max(\sharp S)$ **in ASP.** An ASP program that finds the maximum number of sectors can be simply obtained by considering rules 22–25 together with the optimization statement

```
    #minimize{ empty(S): s(S) }.
```

and the solver will find a sectorization with minimum number of unused (empty) sectors.

Finally, to implement the satisfiability version in ASP in order to test whether $S_{max}(N_v) = S_{max}(N_v - 1) + 1$, instead of using the maximization statement above, we impose that no empty sector exists:

```
:- empty(S).
```

On the complexity of $Max(\sharp S)$. $Max(\sharp S)$ seems very related with the minimum *k-cut* problem [116]. In this problem the aim is to compute the minimum weight multicut that separates the network into k non-empty components. The problem is known to be NP-Hard for an arbitrary $k \geq 3$. However, for fixed k the problem is polynomially solvable in $O(n^{k^2/2-3k/2+4}T(n,m))$, where $T(n,m)$ is the complexity of the min-cut problem on a network of n nodes and m edges.

The $Max(\sharp S)$ maximizes k for a fixed size of the multicut, whereas *k-cut* minimizes the multicut size for a fixed k: the question would be then whether k-cut is the dual of $Max(\sharp S)$. Anyway, the k-cut can not be easily exploited to compute a tight bound on N_s; in fact, to know how many sectors can be drawn with N_v valves it would be necessary that $N_v(k)$ (or $N_v(N_s)$) is also monotonic, and then run the procedure to find out the minimum k-cut several times by increasing k, until the objective value overcomes N_v.

3.6.2 Symmetry Breaking on the Names of Sectors

Given a valve placement, by changing the identifiers of the sectors we obtain a solution that has identical value of objective function (and, indeed, is equivalent from the hydraulic viewpoint) but corresponds to a different solution, as explained in Section 3.4.3; this also happens in the ASP approach. For example, if we have N_s sectors, each valve placement has $N_s!$ equivalent answer sets, which expands super-exponentially the solution space searched by the ASP solver.

A first, simple way to reduce the search space is to impose a naming convention to the sectors containing given edges. For example, we can sort the edges (and assign an integer identifier to each of them) and then state that the sector containing edge 1 must be called sector 1. This already reduces the search space of a factor $1/N_s$. Now, we cannot force the sector containing edge 2 to be called sector 2, because edges 1 and 2 could be in the same sector. So, we impose that the sector containing edge 2 is called either 1 or 2. In the same way, edge i can be in one of the sectors from 1 to i. To program this in MILP the following constraints can be set up:

$$z_i^s = 0, \qquad \forall i \in \overline{V} \mid i > s, s \in S. \tag{3.20}$$

To program the same in ASP, Rule 24 becomes:

```
46 1 { sector(e(X,Y),S): S<=I: index(e(X,Y),I) } 1:- e(X,Y).
```

where predicate `index/2` associates a unique integer identifier to each edge.

Although this strategy reduces significantly the number of symmetries, it does not remove all of them. In particular, edges with index $i \geq N_s$ (where N_s is the maximum number of sectors) are associated to the same set of sector names by the rules 24 and 46. In the most unlucky situation, all the first N_s edges are in the same sector, and this strategy does not remove any symmetry neither in MILP nor in ASP.

To break all symmetries, one can reason as follows. The i-th edge should be either in one of sectors containing the first $i-1$ edges, or in a new sector. So, the i-th edge can belong to the sectors with names from 1 to $\max_{1 \leq j \leq i} S_j$ if S_j is the sector containing edge j. To implement this in MILP the following set of constraints should be imposed:

$$z_i^s \leq \sum_{j \in \overline{V}: j<i} z_j^{s-1}, \forall i \in \overline{V} \mid i > 1, \forall s \in S \mid s > 1$$

This means that if a node i with index I is in the sector `S` then the sector `S-1` contains at least one edge with smaller index.

To implement the same in ASP, the following integrity constraint can be imposed:

```
:- s(S1), S1==S-1,
   index(e(A,B),I), sector(e(A,B),S),
   0 { sector(e(C,D),S1): index(e(C,D),I2): I2<I } 0.
```

This means that if an edge with index `I` is in the sector s then the sector $s-1$ contains at least one edge with smaller index. Note that this technique is applicable to sectors, but not to extended sectors of the ASP program in Section 3.5.2, since an edge might belong to more than one extended sector.

3.6.3 Symmetry breaking on valves positioning

As noticed in [34], if a junction node of the network has cardinality 2 there is no reason neither to i) place two valves around it nor ii) try both positioning of the same valve. The former is pretty clear being the junction nodes weightless. The latter is due to a symmetry, in fact placing a valve on one side of the junction determines the same objective value w.r.t. placing it on the other, being again the junction nodes weightless.

To avoid the solver to search on these choices, the following constraints are included into the MILP formulation:

$$\tau^s_{ij} = 0, \forall (i,j) \in E \mid card(i) = 2, \exists (i,k) \in E, k < j, \forall s \in S$$

and into the ASP programs:

```
:- sym(I,J), sym(I,K), J<K, card(I,2), valve(I,J.
```

where predicate `card` defines the cardinality of node `A`. Both prevent that a valve is installed on a given side of a junction node with cardinality 2.

3.6.4 Redundant valves elimination

As noticed in [34], if a closed path in the graph (i.e., a cycle) contains exactly one valve, then that valve is redundant. In fact, even when the valve is closed, the water can find another path from one side of the valve to the other, so that the valve cannot possibly separate two sectors. So, one can reduce the search space by discarding a priori those valve placements that contain redundant valves.

In MILP this can be implemented by means of the following variables and constraints:

$$\sum_{(i,j)\in C} (\tau^s_{i,j} + \tau^s_{j,i}) \geq 2\tau^s_{k,l} \qquad \forall (k,l) \in C, \forall C \in \mathcal{C}, \forall s \in S$$

$$\sum_{(i,j)\in C} (\tau^s_{i,j} + \tau^s_{j,i}) \geq 2\tau^s_{l,k} \qquad \forall (k,l) \in C, \forall C \in \mathcal{C}, \forall s \in S$$

where C is a cycle in a set of cycles $\mathcal{C}$. This constraint basically states that if there is at least one valve in the cycle, the sum of valves must be greater than 1. However the MILP model for graph partitioning in Section 3.4.1 prevent already to place redundant valves, and there is no reason to impose the constraints above.

In ASP, this optimization amounts to add an integrity constraint:

```
:- cycle(K), 1 { valve(I,J): cycle(K,e(I,J)),
                 valve(J,I): cycle(K,e(I,J)) } 1.
```

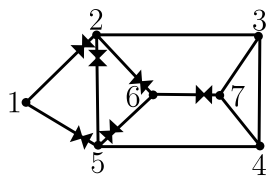


Figure 3.15: Configuration containing a redundant valve

assuming that each cycle (for which one wants to forbid redundant valves) is declared with a fact `cycle(K)` and membership of edges to cycles is declared with facts `cycle(e(A,B),K)`.

The number of cycles is exponential in the size of the graph; in order to reduce the number of constraints of the ground program, one can use a smaller set of cycles. In [34], the cycles considered for redundant valve elimination are *faces* of the graph, that are a concept of planar graphs. When drawing the graph on a plane, each of the regions surrounded by edges of the graph is called a *face*; the number of faces is polynomial, as proven by Euler, and they can be computed in polynomial time. However, hydraulic networks are not planar in general, and the detection of redundant valves can be further improved.

All the cycles in a general graph with loops can be expressed by means of a cycle basis ($\mathcal{CB}$) that is always of size $m-n+1$ having m edges and n nodes [117]. Any other cycle is a linear combination of the vector space of $\mathcal{CB}$. In non-planar graphs the closest concept to the *faces* is the Minimum Cycle Basis (MCB), i.e., the cycle basis that minimizes a function cost on the edges. Considering all unit weights the minimal cycle consists of *chordless* cycles. There are several polynomial algorithms to compute the MCB [118, 119] and the most known are the de Pina [120] and the Horton [117] algorithms.

Figure 3.15 shows a configuration of valves on a toy network that contains a redundant valve, i.e., $v_{2,5}$. This valve is the only within cycle $C = \{2, 3, 4, 5\}$, so it does not split the cycle into different sectors. This redundant valve would not be detected by the faces, since cycle C is not a face of the planar graph. However, it is a chordless cycle and consequently it is included on the minimal cycle basis.

We conjecture that imposing constraints for the elimination of redundant valves on the minimum cycle basis is sufficient to prevent redundant valves on longer cycles; but no proof supports this conjecture yet.

Further remarks. Finally, on propagation based solvers such as ASP solvers, we expect all these strategies can be vary effective. Instead, for MILP the above strategies do not tighten the continuous relaxation, which is the generator of the upper bound on the optimal value. This means that these strategies can possibly fix some variables along the B&B tree, but it is difficult to say whether and how effective they can be. In fact, sometimes adding constraints to a MILP model makes the solving time of the continuous relaxation slightly longer, and more time is spent in every node of the B&B.

Table 3.3 reports the exact approaches for the BIVLP proposed in this discussion, plus the previous CLP(FD) [34]. The table also links these approaches to the features developed to reduce the search space, such as Symmetry Breaking (S.B.) strategies, and redundant valves elimination.

Table 3.3: Approaches and features developed for the BIVLP

Approach / **Feature**		Logic Programming			
			Answer Set Programming		
	MILP	**CLP(FD)**	**Pipe Isolation**	**Extended Sectors**	**Sectors**
S.B. on sectors	✓	n.n.	n.n.	n.a.	✓
S.B. on valves	✓	✓	✓	✓	✓
Redundant valves elimination	n.n.	On faces	MCB	MCB	MCB

n.n.: not needed - n.a.: not applicable - MCB: Minimum Cycle Basis

3.7 Computational results

The experimental platform for the BIVLP (Section 3.1.1) consists of a MILP (Section 3.4) model and 3 ASP encodings (Section 3.5.1) plus a CLP(FD) program [34].

Optimization runs were performed on an *Intel Dual Core* architecture based on *E6550* CPUs with 2.33GHz and 4GB of RAM. Only one core and a timeout of 86400 seconds (24 hours) were used for each run. We used the Gurobi [61] solver v5.0 to solve the MILP model, and the *Clasp* solver, provided by Potsdam Answer Set Solving Collection (Potassco) [114] to solve the ASP programs. The grounding time of *Gringo* for the ASP programs was negligible for all the tested instances.

We run the algorithms on the following three hydraulic networks. *Apulian* (Figure 3.17a) is the benchmark considered in [7,34], and represents a part of the distribution network of the Apulia region in Italy. In particular, studies [7,34] used in this network a limit of at most one valve per pipe. *Realtown* (Figure 3.18a) is the distribution network of a real city (the details are sensitive data and cannot be disclosed). *Anytown* (Figure 3.19a) is a widely used benchmark in the hydraulic engineering literature, and represents the WDS of a typical American town. The ASP programs and the instances are available online.[1] The aim is to analyze the computational behaviour of the various approaches and configurations by varying the networks and the number of available valves. For each network the number of valves ranges in $[N_v^{min}, \ldots, N_v^{max}]$, i.e., from the minimum number of valves that disconnects the network from the sources to the maximum number the network can host.

The MILP model and the ASP programs have been dimensioned on N_s by using the upper bound $UB_{\sharp S}(N_v)$ in Eq. (3.16), and the maximum number of sectors $S_{max}(N_v)$ obtained by the optimisation of the $Max(\sharp S)$ model. For example, in the Apulian network the cardinality of the tank node is 3, so ${N_v}^{min} = 3$ and $UB_{\sharp S}(N_v) = N_v - 2$. Further details about the computational results for $Max(\sharp S)$ will be provided just below.

Computational analysis of $Max(\sharp S)$. As discussed in Section 3.6.1, the size of the search space for the MILP and some of the ASP programs depends on a critical parameter: the maximum number N_s of sectors, for which we proposed in Eq. (3.16) an easy to compute upper bound $UB_{\sharp S}(N_v)$.

However, a tighter bound can be computed with a separate optimization process, as explained in Section 3.6.1, to compute the real maximum number $S_{max}(N_v)$ of sectors in the given distribution network. The resulting optimization process consists now of two steps, namely computing $S_{max}(N_v)$ and minimising the worst service disruption (Figure 3.14); of course, the total computing time, given N_v, should consider both the running times. Also the program $Max(\sharp S)$ needs to be dimensioned on the maximum number of sectors and takes in input $S_{dim}(N_v)$.

To compute it we used the bound given by Corollary 4, so in order to estimate the maximum number of sectors for N_v valves, $S_{max}(N_v)$, we use the information about the maximum number of sectors for $N_v - 1$ valves, $S_{max}(N_v - 1)$. In particular, recall that for Eq. (3.18):

$$S_{max}(N_v) = S_{max}(N_v - 1) + I \qquad\qquad I \in \{0, 1\}$$

[1]`www.ing.unife.it/en/research/ai/vp_asp`

the problem can be considered as a satisfiability problem: finding if with one more valve we can create one more sector (and $I = 1$) or not (and $I = 0$).

Figure 3.16 shows the time for solving both the MILP and ASP implementations of $Max(\sharp S)$ in the Apulian network, varying the number of valves. The same graph shows also $S_{dim}(N_v)$, i.e., the input parameter, and the actual number of sectors $S_{max}(N_v)$. Note that the running time has strong oscillations: it depends on N_v but also on the $S_{max}(N_v)$ value. If the satisfiability problem has a

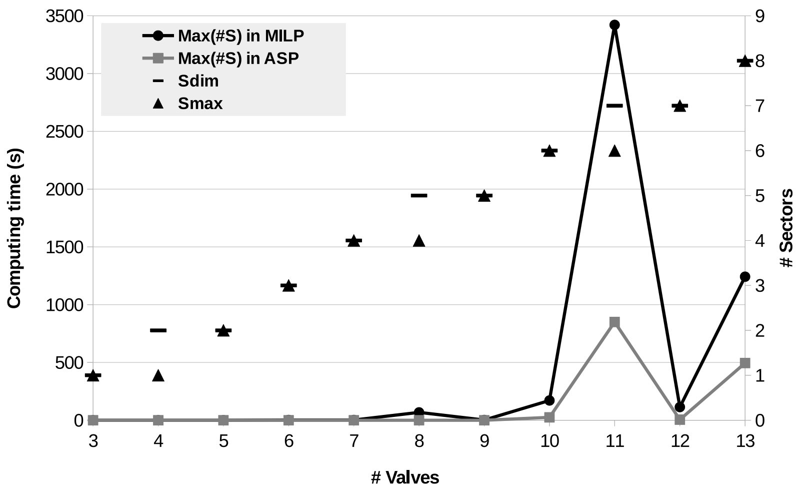


Figure 3.16: $S_{max}(N_v)$ values and computing times of $Max(\sharp S)$ in ASP and MILP for the Apulian network

solution (and $I = 1$), the search can immediately stop after finding it, while if this problem has no solution (and $I = 0$), the solver has to explore the whole search tree to prove that there is no solution for $I = 1$. However, this computing time is almost negligible compared to the time for computing the best valve placement. The ASP implementation has lower computing time than the MILP one.

Computational analysis of the MILP model. Table 3.4 reports the solving time of Gurobi for the MILP model on the Apulian network, with $N_v \in [3, \cdots, 10]$. Up to 8 valves the best configuration is the one that uses $S_{max}(N_v)$ and no Symmetry Breaking (SB) constraints on the sectors (see Section 3.6.2). For 9 and 11 valves the best configuration is again $S_{max}(N_v)$ but with SB. For 10 valves is $S_{max}(N_v)$ without SB. Consequently, tighter bound on N_s can help the search, whereas symmetry breaking constraints on the sectors' names do not give a clear advantage. It is worth mentioning that Gurobi can be configured to perform its

Table 3.4: Solving time (s) of Gurobi with different configurations

$\mathbf{N_v}$	$\mathbf{UB_{\sharp S}(N_v)}$		$\mathbf{Max(\sharp S)}$	$\mathbf{S_{max}(N_v)}$		
	N_s	Time	Best time	N_s	Time	Time (SB)
3	1	0	0	1	0	0
4	2	0	0	1	0	0
5	3	0	0	2	0	0
6	4	**5**	0	3	1	14
7	5	**110**	0	4	61	114
8	6	850	0	4	**267**	460
9	7	2905	0	5	2992	**2055**
10	8	17300	25	6	**9598**	timeout
11	9	timeout	853	6	timeout	**59420**
12	10	timeout	5	7	timeout	timeout

own symmetry breaking strategies; however this always increases solving time than the MILP one.

Computational analysis of the ASP programs. The Pipe Isolation and the Extended Sectors encodings use information about all possible paths between tanks and edges to handle unintended isolations. Between these two programs the Extended Sectors need in general a lower number of comparisons to determine the worst isolation case, so we expect this one to be more effective than Pipe Isolation. Moreover, since the number of paths increases exponentially with the number of edges, we expect in general a better performance of the *Sectors* encoding, which builds up a sectorization and computes reachability of sectors. The charts in Figures 3.17b, 3.18b and 3.19b show that the *Sectors* encoding outperforms the others for all the networks. Also, the Extended Sectors outperforms Pipe Isolation.

A further series of experiments studies the effectiveness of the optimizations shown in Section 3.6, namely the elimination of redundant valves (Section 3.6.4) and the symmetry breaking on sectors' names (Section 3.6.2). In these experiments, we considered only the *Sectors* encoding, that had performed best in the previous set of experiments. In the graphs in Figures 3.17c, 3.18c and 3.19c, we compare the versions of the *Sectors* encoding: the **base** encoding, the addition of **Symmetry Breaking** constraints, the removal of redundant valves based on **Cycles**, and the combination of symmetry breaking with redundant valves elimination (**Complete**). In the *cycles* and *complete* configurations, we consider the elimination of redundant valves based on **all** cycles or only on **minimal** cycles. As we can see, in all networks both the symmetry breaking and the redundant valve elimination provide a notable speedup; moreover, in general combining the two techniques provides a further improvement.

The number of cycles used in the redundant valves elimination has also an effect on the computing time. In some cases considering all the cycles (instead of only the minimal ones) provides some improvement, while in others increasing the number of cycles increases the computing time. It is worth noting that in all cases, considering all cycles is better than considering no cycles at all, however, in general, removing the redundant valves only on the minimal cycles provides better performance. This can be seen in particular in the “Anytown” benchmark: in this case the number of cycles is very large (15267), so the resulting ground program can be very large for the solver.

In Figures 3.17d, 3.18d and 3.19d we compare the running time necessary to find the best valve placement with different estimations for the upper bound on the number of sectors: either setting $UB_{\sharp S}(N_v)$ (for brevity $S_{dim} = UB(\sharp S)$), or using the two-stage architecture (Figure 3.14), where the first phase computes the maximum number of sectors and the second feeds the value $S_{max}(N_v)$ as input to the optimal valve placement (for brevity $S_{dim} = Max(\sharp S)$). The second column in Figures 3.17d, 3.18d and 3.19d provides always the best running time, even though it is the sum of two optimization processes. The timings of the two processes are shown in the graphs by splitting the column in two; the contribution of the first phase is almost negligible.

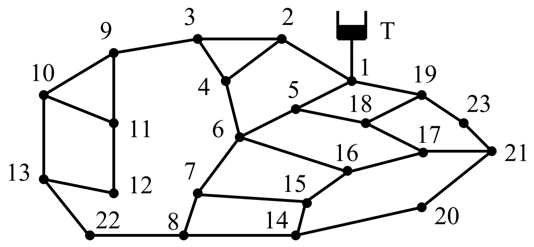
Finally, Figures 3.17e, 3.18e and 3.19e show that the CLP(FD) formulation is still faster than the best configuration of the *Sectors* ASP encoding (i.e., the *Complete(min)* configuration with $S_{dim} = max(\sharp S)$). Both our MILP and ASP formulation are not able to outperform CLP(FD) yet, but they can be surely improved and also pave the way to hybrid and metaheuristics approaches. Next section poses practical insights about further research on the optimal placement of isolation valves.

Computational comparison between MILP and ASP approaches. Table 3.5 reports computing times of the ASP *Sectors* and the MILP programs for the Apulian network. The MILP program outperforms the ASP one for 10 valves in the base version (without symmetry breaking). It also outperforms the ASP version with SB but without MCB for 11 valves. However, the best configuration among the ASP and the MILP versions is the ASP with SB and MCB.

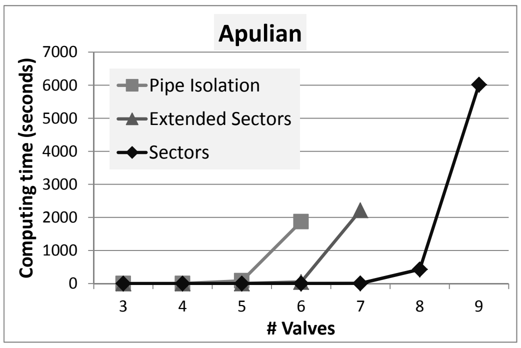
These results show that symmetries have a notable role in the effectiveness of the solving algorithms, both in MILP and ASP. The additional symmetry breaking constraints discussed in Section 3.6 reduce the computing time for the ASP programs, but they are unhelpful for the MILP solver. This suggests to design ASP encodings that assign the pipes to the sectors through deterministic rules, so that no symmetries are introduced. In MILP, symmetry breaking should be performed by implementing orbitopal fixing [97] into the search tree rather than by cabling hard constraints into the model.

Table 3.5: Computing time (s) of the ASP encoding *Sectors* and the MILP program using N_s^* for the Apulian network

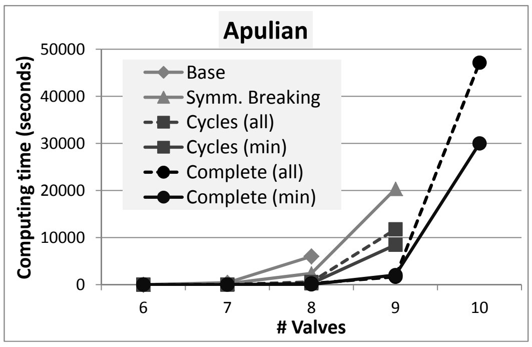
	Base		**SB**		
$\mathbf{N_v}$	**ASP**	**MILP**	**ASP**	**ASP + MCB**	**MILP**
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	1	0	0	14
7	7	61	4	**1**	114
8	67	267	29	**4**	460
9	1016	2992	350	**51**	2055
10	51722	9598	17703	**889**	timeout
11	timeout	timeout	timeout	**8137**	59420
12	timeout	timeout	timeout	timeout	timeout



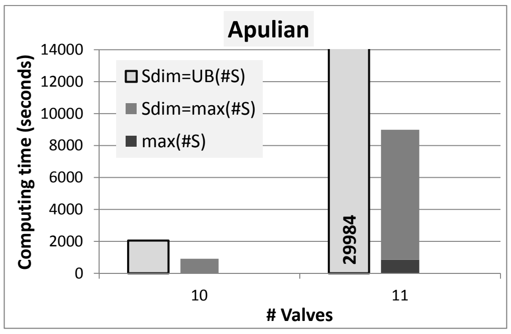
(a) Apulian WDS



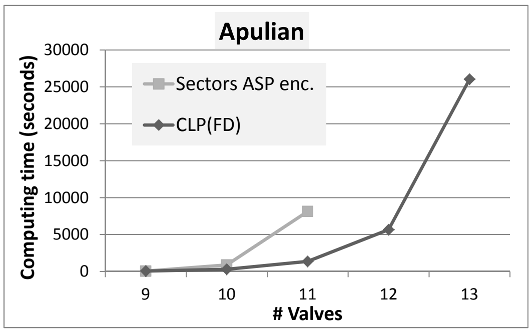
(b) Encodings comparison



(c) *Sectors* configuration comparison

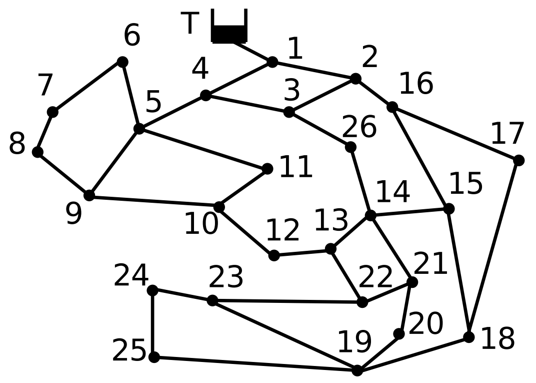


(d) S_{dim} dimensioning impact

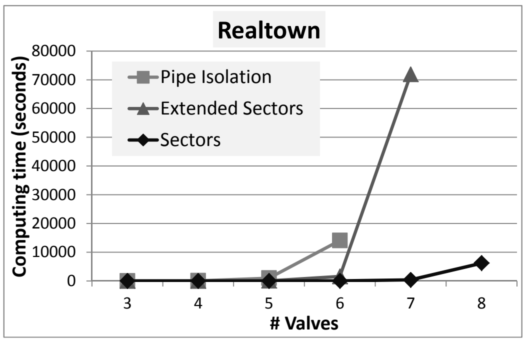


(e) Comparison with CLP

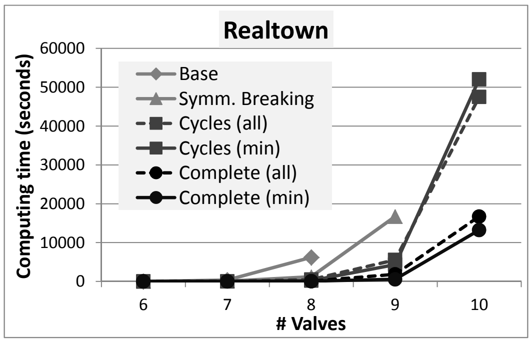
Figure 3.17: The Apulian network and optimisation performance



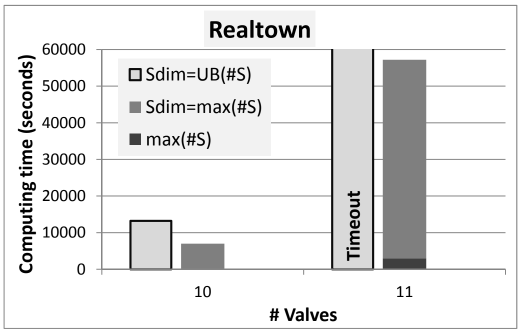
(a) Realtown WDS



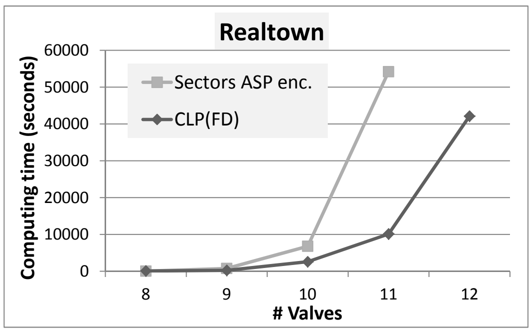
(b) Encodings comparison



(c) *Sectors* configuration comparison

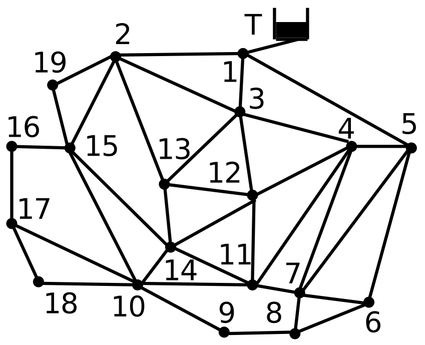


(d) S_{dim} dimensioning impact

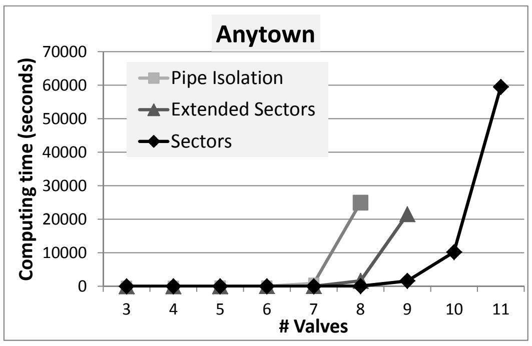


(e) Comparison with CLP

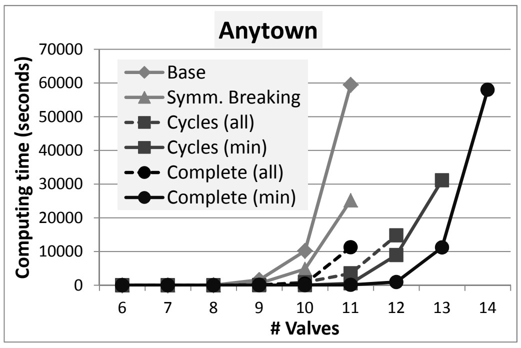
Figure 3.18: The Realtown network and optimisation performance



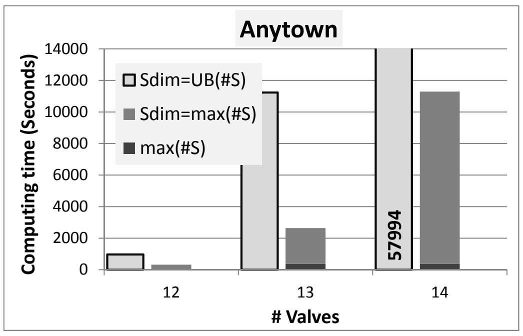
(a) Anytown WDS



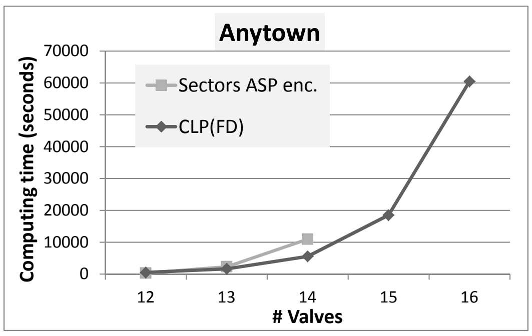
(b) Encodings comparison



(c) *Sectors* configuration comparison



(d) S_{dim} dimensioning impact



(e) Comparison with CLP

Figure 3.19: The Anytown network and optimisation performance

3.8 Future improvements in IVLP

Experimental results (Section 3.7) suggest to investigate further on MILP and ASP approach, being computationally weak w.r.t. the CLP behaviour. Several ideas to improve the state of the art are discussed along the next pages.

3.8.1 Further research on the MILP model

The MILP model has been solved by Branch and Bound; unfortunately the continuous relaxation does not provide good bounds on the satisfied demand (see Section 3.4.3), thus the search tree is not pruned effectively. Also, symmetries may even enlarge the search, but cabling symmetry breaking constraints into the model does not give clear advantages.

To improve the MILP approach the following points should be taken into account:

- tighten the continuous relaxation;
- improve the upper bound by means of heuristics;
- integrate ad hoc branching strategies;
- handling symmetries.

A bounding strategy for the BIVLP was proposed in [34]; even though it is based on the internal demands of the sectors it can be performed on some nodes of the B&B tree. Unfortunately it is not suitable to catch also unintended isolations.

Ad hoc branching strategies can be designed by considering topological aspects of the network. For example the brancher can give higher priority to the variables of the current biggest sector. The search can also use the cycles to select the variables; in fact whenever a cycle contains only one valve, the branching priority increases on the other variables of the cycle.

Symmetry breaking can be integrated into the search rather than into the model. Orbitopal fixing [97] is a linear procedure that can be performed into any node of the search tree for "partitioning orbitopes". A partitioning orbitope is the polytope drawn by a matrix of binary variables b_{ij} so that:

- each row is linked by a cardinality constraint of the form $\sum_i b_{ij} = 1$, and
- the columns are sorted in a decreasing lexicographical order.

It is actually the case of the partitioning variables z^s, that are involved into the partitioning constraints in Eq. (3.2a) and into the ordering constraints in Eq. (3.20). Instead of adding additional constraints, this procedure checks whether same variables are in the same orbitope of some other that are already fixed; then, it fixes them simultaneously. This strategy had performed well for graph partitioning problems and it can be promising for the BILVP either.

A Benders decomposition approach for IVLP

Recall the MILP model in (3.9a–3.9d):

$$\begin{aligned}
&\max \Delta && && (3.9a)\\
&\quad s.t. && && \\
&\qquad GP(\tau), && && (3.9b)\\
&\qquad \Delta \le x^s_{P,\sigma} && \forall\ s \in S, && (3.9c)\\
&\qquad FP(\tau^s) && \forall\ s \in S. && (3.9d)
\end{aligned}$$

Let $\overline{\tau}$ be a feasible assignment for the set of binary valves τ; then the problem becomes:

$$\begin{aligned}
&\max \Delta && && (3.22a)\\
&\quad s.t. && && \\
&\qquad \Delta \le x^s_{P,\sigma} && \forall\ s \in S, && (3.22b)\\
&\qquad FP(\overline{\tau^s}) && \forall\ s \in S. && (3.22c)
\end{aligned}$$

This may seem a Maximum Concurrent Flow Problem [121] at first sight, that is NP-Hard. But notice that the s-th flow does not share the same network of the others, so the flows of different sectors do not contribute to the same capacity. Consequently the problem in (3.22a–3.22c) is a common Maximum Flow Problem on parallel networks; to be solved an independent optimisation can be performed for each sector. At the end the minimum-maximum flow is the optimal value of δ. This algorithms runs in polynomial time, namely $O(sF(m,n))$, where $F(m,n)$ is the complexity of the algorithm used to maximise the single flow problem on a network with m edges and n nodes.

This means that fixing the *hard* variables τ and z makes the problem easy to solve. Typically in these scenarios a Benders Decomposition [24, 122] can be applied. The problem to assign the hard variables is called the *master* problem, and the easier nested problems are called *subproblems*. If the subproblems are linear programs the decomposition is the classical one from Benders, otherwise, it is called the Generalized Benders Decomposition [123, 124].

The master problem in this case would be:

$$\max y \tag{3.23a}$$
$$s.t.$$
$$GP(\tau), \tag{3.23b}$$
$$c(\tau^s) \geq y \qquad \forall k \in K, s \in S \tag{3.23c}$$
$$c(\tau^s) \geq 0 \qquad \forall j \in J, s \in S \tag{3.23d}$$

where Eq. (3.23c) and (3.23d) are called "Benders cut"; the former is on *optimality* cut and it is obtained from the dual of the subproblem every time it has an optimal solution; instead the latter is a *feasibility* cut and it is obtained every time the subproblem is infeasible. In particular K is the set of feasible suproblems, and J is the set of infeasible subproblems. The dual of the subproblem in this case would be Minimum Cut Problems as formalized in (3.11a–3.11l). So, accordingly with (3.11a) the shape of the Benders cut would be:

$$\begin{aligned} c(\tau^s) \equiv & \sum_{(i,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Upsilon(1-\tau^s_{ij})(\omega^s_{i,\epsilon_{ij}} + \omega^s_{\epsilon_{ij},i}) + \\ & + \sum_{(j,\epsilon_{ij})\in\overline{\overline{\Psi}}} \Upsilon(1-\tau^s_{ji})(\omega^s_{j,\epsilon_{ij}} + \omega^s_{\epsilon_{ij},j}) + \\ & + \sum_{(\epsilon_{ij},P)\in\overline{\overline{\Pi}}} (\delta_{\epsilon_{ij}}\omega^s_{\epsilon_{ij},P}) \end{aligned} \tag{3.24}$$

Notice that in (3.24) ω are not variables anymore, but coefficients.

This procedure continues until the subproblem and the master problem have the same objective value. In the worst case an exponential number of Benders cuts are needed to achieve the optimality; however, the Benders decomposition has been applied to many constrained optimisation problems on networks [125].

The master level is still an integer program and needs to be solved by B&B, so symmetry breaking by orbitopal fixing could be useful either.

Finally, the optimality cuts in this application contain only integer coefficients. This makes us interested into investigating hybrid resolution of the Benders decomposition by means of ASP (the master), which better handles symmetries, and linear programming (the subproblems). Unfortunately, for feasibility cuts the integrality property does not hold anymore. However we believe that the subproblems can not be infeasible being the configuration of valves always feasible. This should be investigated either.

3.8.2 Further research on the ASP model

This work proposes also three encodings in Answer Set Programming. The first encoding keeps the sectors implicit; for each computes a configuration of closed valves and compares the satisfied demand of any isolation case with all the others. The second also starts from the configuration of closed valves, groups the isolated pipes into sectors, and compares only the unsatisfied demand of sectors. The third encoding installs the valves, assigns the sectors to the pipes and compares the unsatisfied demand between every pair of sectors.

As already mentioned in Section 3.5.2, both the first and the second encodings have a main weakness: the solver for each pipe has to guess on the closed valves; this makes the solver wrong every time the union of the closed valves is not a feasible placement of the valves, and a lot of choices turn out as infeasible. The encoding based on the sectors installs the valves globally, but then it also assigns pipes to the sectors, begetting a huge amount of symmetry, though symmetry breaking can be effectively imposed by some constraints.

A better ASP encoding would place the valves globally as in the third encoding, would group pipes into sectors in a deterministic way (without delegating to the search) and finally would compare the unsatisfied demand of the sectors through the lowest number of comparisons. A version of the ASP encoding implementing these ideas was proposed at the Model & Solve track of the ASP competition 2013, where the BIVLP was part of the benchmark set. [2]

3.8.3 Hybrid metaheuristics approaches

A hydraulic network can be sketch out in order to represent only the main skeleton of the infrastructure. Thus, sometimes small networks can represent very big infrastructures and a valve placement is computed for that simplified layout. However even simplified layouts can be made of hundreds of pipes; in this case no exact approaches in the literature are scalable enough to find even feasible solutions. Neither metaheuristic approaches up to now faced with network of such a size.

The most difficult task in big network would be represent (and find) a feasible sectorization. For example, the genetic algorithm in [7] encodes each possible position of a valve with a boolean variable; so the length of the encoding is $2m$ with m pipes and the cardinality of the search space is $2m!$. Hydraulic networks are typically highly connected, which means they often contain a lot of loops (or cycles). As stated in Section 3.6.4 the size of a cycle basis is $m-n+1$, and feasible

[2]`https://www.mat.unical.it/aspcomp2013/`

positioning of valves would never place a single valve on a cycle of the minimal cycle basis. A possible genetic encoding would map the placement of the valves on the minimal cycle rather than on the feasible positions. For example the gene may state whether on that cycle there are valves or not, otherwise it may state how many valves it contains. Similarly to the architectures in Chapter 2, in this case the genetic algorithm needs to be coupled to submodules in order to:

- compute new feasible individuals;
- compute the fitness of the solution.

The former is needed to prevent the exploration of infeasible solutions. The latter is needed to find the actual placement of the valves given the individual's DNA. Notice that on big instances the second point would be an optimisation with an important amount of fixed values; since cycles can be easily represented in ASP, we believe that submodules of this architecture could be written in ASP.

Conclusions

In this work, two real applications in hydroinformatics have been addressed, namely:

- the *Response to Contamination Problem* (RCP), an operative optimisation problem, and
- the *Isolation Valves Location Problem* (IVLP), a strategic optimisation problem.

These problems were addressed in the hydraulic literature by Genetic Algorithms (GAs) [15, 15, 16]. Our approaches exploit GAs, as well as Local Search (LS), Mixed Integer Linear Programming (MILP), Path Relinking (PR), Answer Set Programming (ASP). To do that we modelled the problems as Constrained Optimisation Problems (COPs) and, in collaboration with the hydraulic engineers, we formalized the constraints in a logical and mathematical fashion. Some of the architectures proposed in this study are hybrid; for example, for the RCP they integrate GA, MILP, and PR. Also, the mathematical model proposed hereby for the IVLP allows for integrating ASP and MILP into the same solving architecture. More details about these results are given just below.

The RCP is the problem to compute the optimal scheduling of operations that the technicians have to implement in case of contamination events of the hydraulic network. The case study was the hydraulic network of Ferrara (Italy), a city of 120, 000 citizens; here 3 teams of technicians were available, and 13 hydraulic devices (hydrants and valves) were selected to be dispatched in response to the contamination scenarios [15]. The aim was to compute the activation schedules of the devices that minimize the volume of contaminated water consumed by the users during the contamination event. The evaluation of this volume given a feasible scheduling of response operations requires a hydraulic simulation, i.e., the objective function is a black box. Every simulation takes about 5 seconds in EPANET [18] (an open–source hydraulic simulator), so the maximum number of available simulations was limited to 500. The problem has been modelled as a

Constrained Optimisation Problem: the technicians have to move on the street layer of the city, so the scheduling should be feasible according to the travelling times and the constraints of the Multiple Travelling Salesman Problem (mTSP); a Mixed Integer Linear Programming (MILP) description for the mTSP has been provided and was inspired by the literature [36]. This is also the first novelty of this work, in fact this model allows for considering all the possible feasible schedules with the given number of teams; other studies in the hydraulic literature assign a team to each device, otherwise if a lower number of teams is available they define by hands some "reasonable" schedules [15,16]. A MILP solver cannot minimize directly the volume of contaminated water, because of the complexity of the physics laws governing the water flows; unfortunately, objective functions interpreting common sense criteria (e.g., doing everything as soon as possible) provided worse solutions with respect to a random search on the feasible solutions. This was the second important result, because in case of emergency a fast response is intuitively a good strategy, whereas in RCP doing randomly is better. Consequently, a *simulation–optimisation* approach has been proposed, integrating GAs, MILP, and EPANET. Two genetic encodings were inspired by the literature of the mTSP [46]; they represent the routes of the teams and performed better than the random search. A novel genetic encoding has been proposed to the RCP; it uses the vector of activation times as chromosome and was inspired by the fact that EPANET takes in input that very vector. The fitness was evaluated by EPANET indeed. Moreover, the latter GA was hybrid because it was linked to a MILP solver to ensure the feasibility of the solutions. The hybrid GA performed better than the literature inspired ones. Almost 3 years of CPU time have been used up to test our methods. The results had been validated by non–parametric statistical tests [63]. Another novelty of this work was the application of Path Relinking (PR) [68] to a mTSP variant and to simulation–optimisation scenarios in general. PR is a well known strategy in Operations Research to improve the solutions coming from another search strategy, and it has been applied to perform an intensification on the final populations of different GA runs. The results encourage further investigations about a tighter integration of PR strategies into the architecture. Finally, the solutions computed by these architectures are better on average than the ones in the hydraulic literature; in particular, the best schedule proposed in [15] was computed for 4 teams, and the averaged volume of contaminated water consumed by the users on 42 contamination scenarios was $44,287$ litres, whereas our hybrid GA, using only 3 teams, computes solutions that are on average lower than $34,000$ litres.

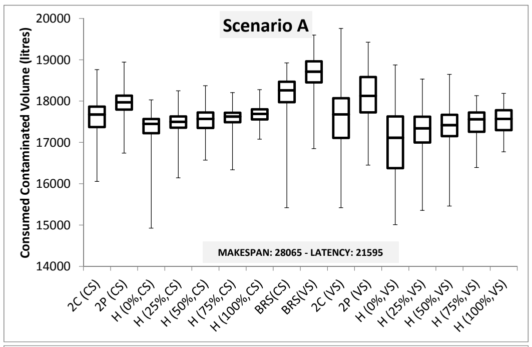
The IVLP is the problem to compute the optimal placement of isolation valves on the hydraulic network. Every time a pipe gets damaged, technicians first isolate it by closing some surrounding valves, only then they fix the pipe. During these operations all the users that are linked to the isolated region of the network experience the lack of water. Every pipe may get broken, thus any part of the city could experience a service disruption for a while, sooner or later. The aim is then to minimize the unsatisfied demand due to the isolations; this would be surely achieved by placing as many valves as needed to isolate always just the broken pipe. Unfortunately, the valves have maintenance and installation costs, so their number is limited; thus, a feasible placement of these valves should determine a *sectorization* of the network. The problem apparently is a common Graph Partitioning Problem, but isolating a part of the network may determine some other *unintended isolations*, and this should be taken into account in order to compute correctly the unsatisfied demand. Hydraulic engineers optimise the placement of the available valves by means of GAs [7, 8], seeking a compromise between cost and solution quality; the former is considered to be proportional to the number of installed valves, the latter is the unsatisfied demand in the worst isolation case. The problem to minimize the worst unsatisfied demand by varying the isolation scenario is called Bottleneck IVLP (BIVLP). The first exact approach for the BIVLP was in Constraint Logic Programming [34], so in Artificial Intelligence; it was also the first study addressing this problem as a constrained problem, and improved the state of the art by computing the optimal solutions that the previous works did not provide. Instead, the first mathematical model for the IVLP was proposed hereby; the model is expressed in MILP and identifies two main problem structures: a main layer that contains the decision variables for the valve placement and the graph partitioning's constraints, and a nested maximum flow layer that contains the variables and the constraints to compute the correct unsatisfied demand of any isolation case. Since the variables in the GP layer are integer and the remaining flow variables are continuous, this formulation paves the way for a Benders decomposition, which will be investigated in future. The MILP model has been also adapted to the BIVLP, and solved by using common techniques in Operations Research. Some weaknesses have been detected in this approach; the MILP model has not a good relaxation and also determines some symmetries. Also, we proposed another approach in Artificial Intelligence, namely in Answer Set Programming. Several ASP programs have been developed for the BIVLP, some compute the unsatisfied demand exploiting the reachability of the pipes from the sources, others define the sectors and their reachability. Defining the sectors by their name yield symmetries also in ASP, but in this case they can be effectively handled by imposing additional constraints, whereas in MILP it was unhelpful. Both the approaches can be surely improved. For example ad hoc symmetry breaking techniques for partitioning problems [97] can be implemented in this case. The ASP programs can be modified in order to make the assignment of pipes to sectors deterministic; this would also remove all symmetries. Finally, since the best ASP programs outperforms the MILP one, the main level of the Benders decomposition could be solved in ASP; to do that it would be necessary to ensure that the cuts coming from the nested level do not contain real numbers.

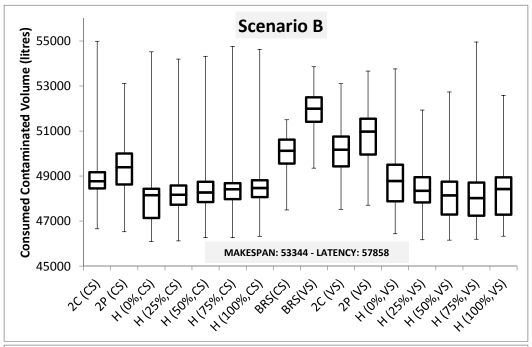
To conclude, detecting and formalizing the constrained structure of the above real problems in hydroinformatics has made possible to i) compute applicable solutions, ii) exploit graph theory for modellization and solving purposes, iii) solve the problem by well suited technologies in Operations Research and Artificial Intelligence, and iv) propose new integrated architectures for a more effective solving. In general, Operations Research and Artificial Intelligence, together with graph theory, provide very well suited and well known theoretical and practical techniques to address constrained optimisation problems. Optimisation issues in hydroinformatics quite often present constrained structures, and we hope this work can also help to give the insight about how strategic and promising the role of these constraints can be.

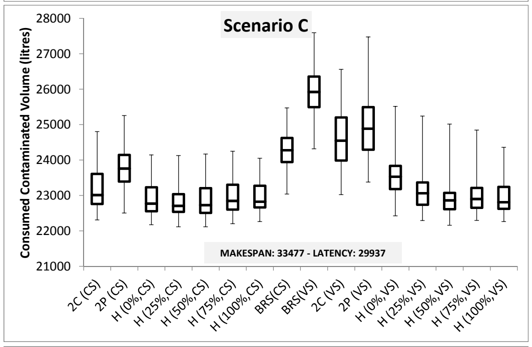
Appendices

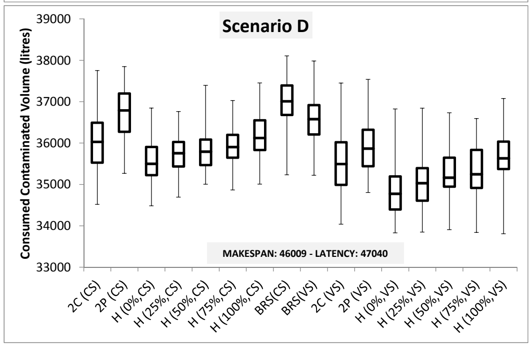
Boxplots

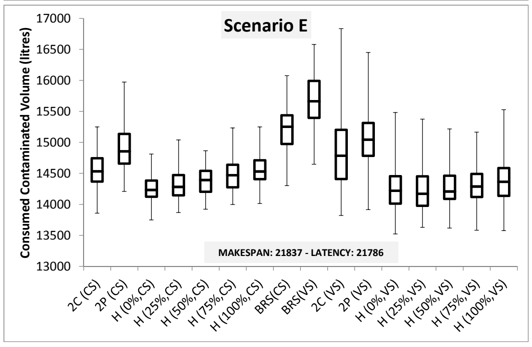
In this section, the result dataset of each scenario is depicted as a boxplot chart. Axis x report GAs and BRSs methods, and axis y report the consumed contaminated volume in litres. Each boxplot involves 100 independent runs. Values of Makespan and Latency solutions are also reported. Boxplots from scenario A to scenario T are listed in alphabetical order.

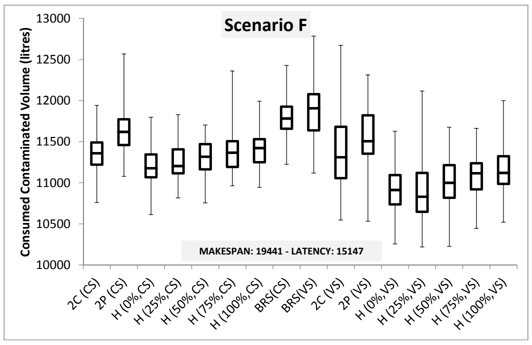


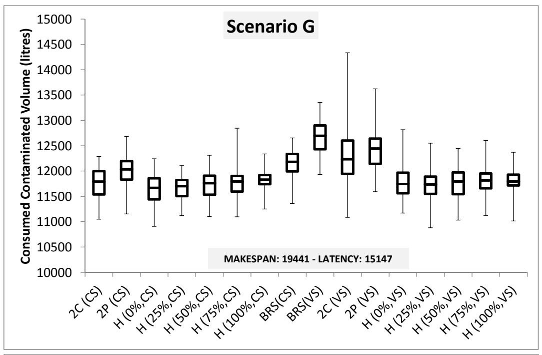


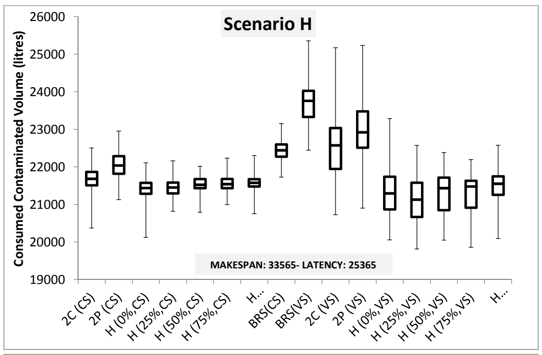


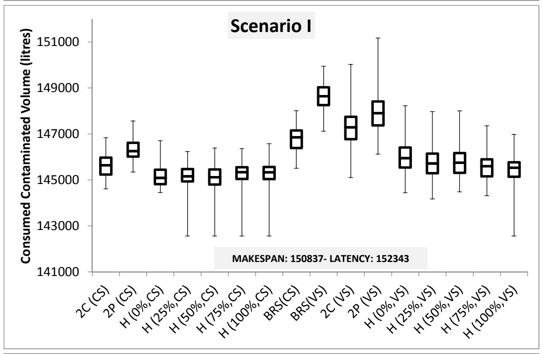


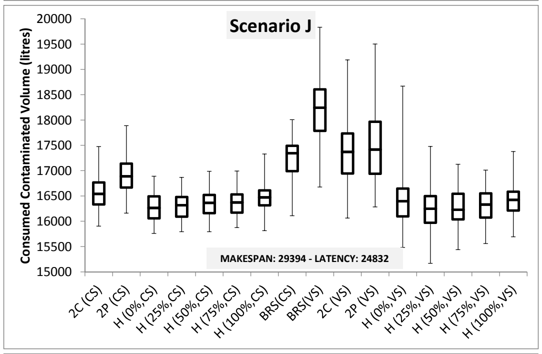


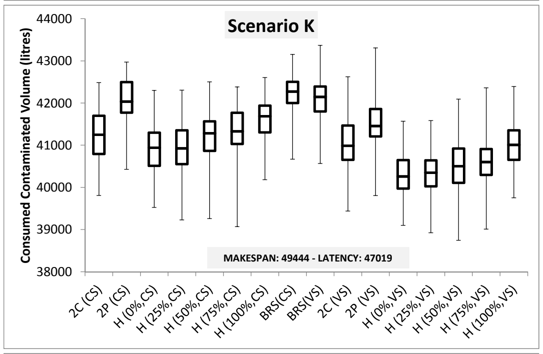


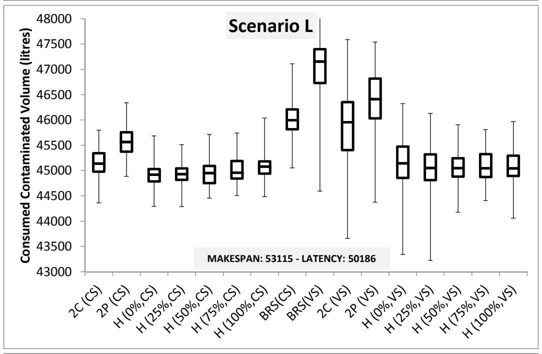

Scenario G
Consumed Contaminated Volume (litres)
MAKESPAN: 19441 - LATENCY: 15147
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
BRS(CS)
BRS(VS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)

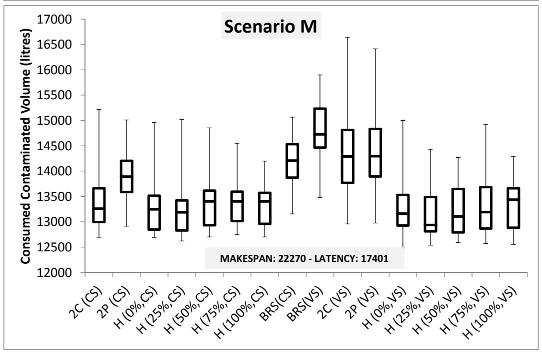

Scenario H
Consumed Contaminated Volume (litres)
MAKESPAN: 33565- LATENCY: 25365

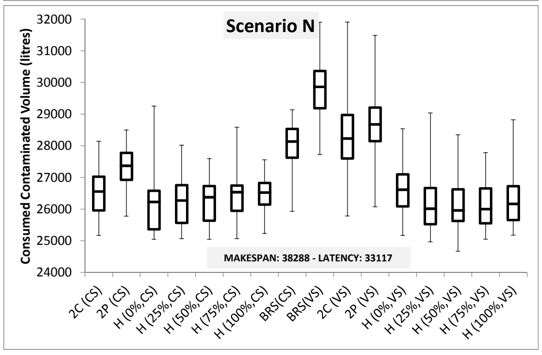

Scenario I
Consumed Contaminated Volume (litres)
MAKESPAN: 150837- LATENCY: 152343

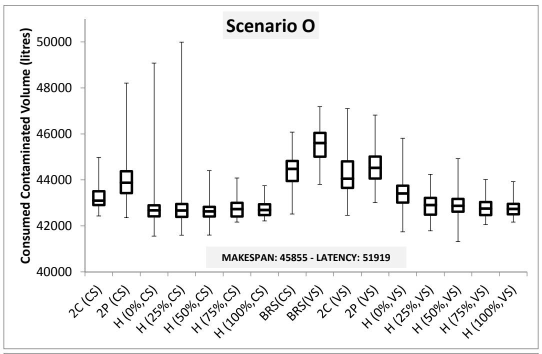

Scenario J
Consumed Contaminated Volume (litres)
MAKESPAN: 29394 - LATENCY: 24832

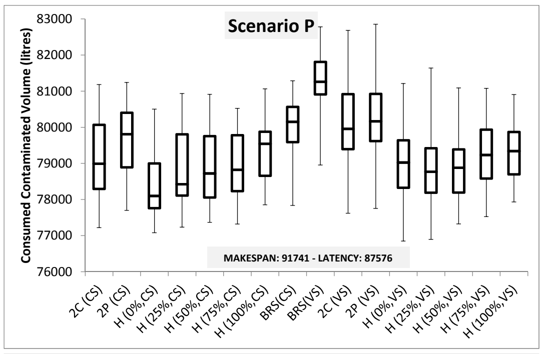

Scenario K
Consumed Contaminated Volume (litres)
MAKESPAN: 49444 - LATENCY: 47019

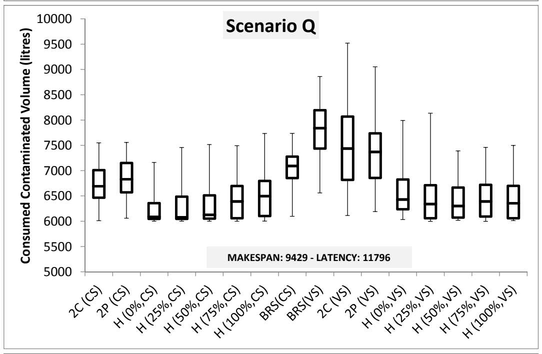

Scenario L
Consumed Contaminated Volume (litres)
MAKESPAN: 53115 - LATENCY: 50186

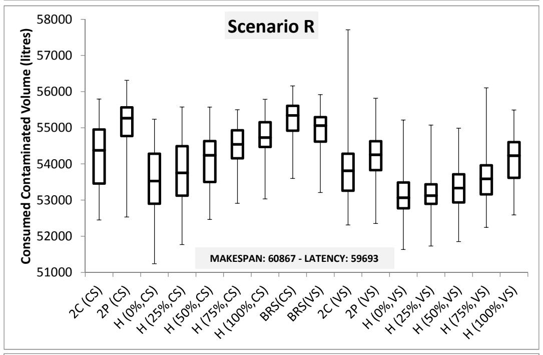

Scenario M
Consumed Contaminated Volume (litres)
MAKESPAN: 22270 - LATENCY: 17401

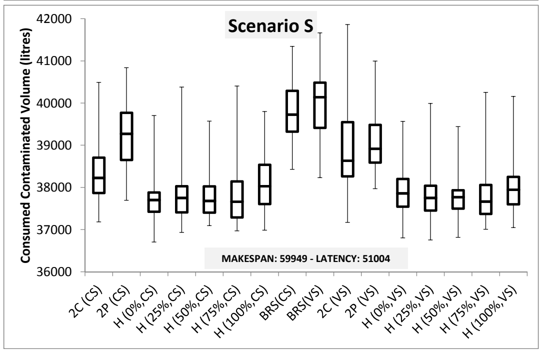

Scenario N
Consumed Contaminated Volume (litres)
MAKESPAN: 38288 - LATENCY: 33117

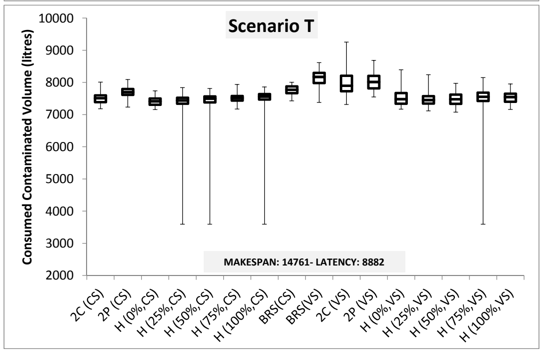

Scenario O
Consumed Contaminated Volume (litres)
MAKESPAN: 45855 - LATENCY: 51919
2C (CS)
2P (CS)
H (0%,CS)
H (25%,CS)
H (50%,CS)
H (75%,CS)
H (100%,CS)
BRS(CS)
BRS(VS)
2C (VS)
2P (VS)
H (0%,VS)
H (25%,VS)
H (50%,VS)
H (75%,VS)
H (100%,VS)


Scenario P
Consumed Contaminated Volume (litres)
MAKESPAN: 91741 - LATENCY: 87576


Scenario Q
Consumed Contaminated Volume (litres)
MAKESPAN: 9429 - LATENCY: 11796


Scenario R
Consumed Contaminated Volume (litres)
MAKESPAN: 60867 - LATENCY: 59693


Scenario S
Consumed Contaminated Volume (litres)
MAKESPAN: 59949 - LATENCY: 51004


Scenario T
Consumed Contaminated Volume (litres)
MAKESPAN: 14761- LATENCY: 8882

Bibliography

[1] E. Todini, “Looped water distribution networks design using a resilience index based heuristic approach,” *Urban Water*, vol. 2, no. 2, pp. 115 – 122, 2000, developments in water distribution systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1462075800000492

[2] E. Creaco and M. Franchini, “Fast network multi-objective design algorithm combined with an a posteriori procedure for reliability evaluation under various operational scenarios,” *Urban Water Journal*, vol. 9, no. 6, pp. 385–399, 2012. [Online]. Available: http://dx.doi.org/10.1080/1573062X.2012.690432

[3] M. Cunha and J. Sousa, “Water distribution network design optimization: Simulated annealing approach,” *Journal of Water Resources Planning and Management*, vol. 125, no. 4, pp. 215–221, 1999. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(1999)125:4(215)

[4] H. R. Maier, A. R. Simpson, A. C. Zecchin, W. K. Foong, K. Y. Phang, H. Y. Seah, and C. L. Tan, “Ant colony optimization for design of water distribution systems,” *Journal of water resources planning and management*, vol. 129, no. 3, pp. 200–209, 2003. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-9496(2003)129:3(200)

[5] M. Franchini, “Un excursus sugli algoritmi per la progettazione e la riabilitazione delle reti di distribuzione idrica,” *L’Acqua*, vol. 2, pp. 15 – 22, 2010.

[6] I. Narayanan, V. Sarangan, A. Vasan, A. Srinivasan, A. Sivasubramaniam, B. Murt, and S. Narasimhan, “Efficient booster pump placement in water networks using graph theoretic principles,” in *Green Computing Conference (IGCC), 2012 International*, June 2012, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6322271

[7] O. Giustolisi and D. A. Savić, “Optimal design of isolation valve system for water distribution networks,” pp. 1–13. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/41024%28340%2931

[8] E. Creaco, M. Franchini, and S. Alvisi, “Optimal placement of isolation valves in water distribution systems based on valve cost and weighted average demand shortfall,” *Water Resources Management*, vol. 24, no. 15, pp. 4317–4338, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11269-010-9661-5

[9] J. J. Huang, E. A. McBean, and W. James, *Multi-objective Optimization for Monitoring Sensor Placement in Water Distribution Systems*, ch. 112, pp. 1–14. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/40941%28247%29113

[10] S. Rathi and R. Gupta, “Sensor placement methods for contamination detection in water distribution networks: A review,” *Procedia Engineering*, vol. 89, no. 0, pp. 181 – 188, 2014, 16th Water Distribution System Analysis Conference, {WDSA2014} Urban Water Hydroinformatics and Strategic Planning. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705814022905

[11] W. E. Hart and R. Murray, “Review of sensor placement strategies for contamination warning systems in drinking water distribution systems,” *Journal of Water Resources Planning and Management*, vol. 136, no. 6, pp. 611–619, 2010. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000081

[12] L. E. Ormsbee and K. E. Lansey, “Optimal control of water supply pumping systems,” *Journal of Water Resources Planning and Management*, vol. 120, no. 2, pp. 237–252, 1994. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-9496(1994)120:2(237)

[13] R. Farmani, G. A. Walters, and D. A. Savic, “Trade-off between total cost and reliability for anytown water distribution network,” *Journal of Water Resources Planning and Management*, vol. 131, no. 3, pp. 161–171, 2005. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-9496(1994)120:2(237)

[14] F. Martinez, V. Hernandez, J. Alonso, Z. Rao, and S. Alvisi, “Optimizing the operation of the valencia water-distribution networ,” *Journal of*

Hydroinformatics, vol. 9, no. 1, pp. 65–78, 2007. [Online]. Available: http://www.iwaponline.com/jh/009/jh0090065.htm

[15] 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*, vol. 6, no. 2, pp. 115–135, 2009. [Online]. Available: http://dx.doi.org/10.1080/15730620802566836

[16] 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*, vol. 136, no. 1, pp. 48–58, 2010. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(2010)136:1(48)

[17] MATLAB, *version R2014b*. Natick, Massachusetts: The MathWorks Inc., 2014. [Online]. Available: http://uk.mathworks.com/products/matlab/

[18] L. A. Rossman, *EPANET 2 users manual*, National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, USA., 2000. [Online]. Available: http://nepis.epa.gov/Adobe/PDF/P1007WWU.pdf

[19] J. April, F. Glover, J. P. Kelly, and M. Laguna, "Simulation-based optimization: Practical introduction to simulation optimization," in *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation*, ser. WSC '03. Winter Simulation Conference, 2003, pp. 71–78. [Online]. Available: http://dl.acm.org/citation.cfm?id=1030818.1030830

[20] S. Shan and G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and Multidisciplinary Optimization*, vol. 41, no. 2, pp. 219–241, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00158-009-0420-2

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[22] D. Jungnickel and T. Schade, *Graphs, networks and algorithms*. Springer, 2008.

[23] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff, *Combinatorics and graph theory.* Springer, 2008, vol. 2.

[24] R. Martin, *Large Scale Linear and Integer Optimization: A Unified Approach.* Springer US, 1999.

[25] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier Science, 2006.

[26] J. W. Lloyd, *Foundations of Logic Programming; (2Nd Extended Ed.).* New York, NY, USA: Springer-Verlag New York, Inc., 1987.

[27] J. Jaffar and M. J. Maher, "Constraint logic programming: a survey," *The Journal of Logic Programming*, vol. 1920, Supplement 1, no. 0, pp. 503 – 581, 1994, special Issue: Ten Years of Logic Programming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0743106694900337

[28] M. Gelfond, "Answer sets," in *Handbook of Knowledge Representation*, F. van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier Science, 2008, ch. 7, pp. 285–316.

[29] K. Ghedira, *Constraint Satisfaction Problems: CSP Formalisms and Techniques*, ser. FOCUS Series. Wiley, 2013.

[30] A. Biere, *Handbook of Satisfiability*, ser. Frontiers in artificial intelligence and applications. IOS Press, 2009.

[31] C. Bragalli, C. DAmbrosio, J. Lee, A. Lodi, and P. Toth, "On the optimal design of water distribution networks: a practical minlp approach," *Optimization and Engineering*, vol. 13, no. 2, pp. 219–246, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11081-011-9141-7

[32] J. W. Berry, W. E. Hart, C. A. Phillips, and J.-P. Watson, *A Facility Location Approach to Sensor Placement Optimization*, 2008, ch. 110, pp. 1–4. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/40941%28247%29111

[33] M. Propato, "Contamination warning in water networks: General mixed-integer linear models for sensor location design," *Journal of Water Resources Planning and Management*, vol. 132, no. 4, pp. 225–233, 2006. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(2006)132:4(225)

[34] M. Cattafi, M. Gavanelli, M. Nonato, S. Alvisi, and M. Franchini, "Optimal placement of valves in a water distribution network with CLP(FD)," *Theory and Practice of Logic Programming*, vol. 11, no. 4-5, pp. 731–747, 2011. [Online]. Available: http://arxiv.org/abs/1109.1248

[35] 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*, vol. 39, no. 1, pp. 57–68, 2009. [Online]. Available: http://pubsonline.informs.org/doi/abs/10.1287/inte.1080.0415

[36] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209 – 219, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048304001550

[37] 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*, vol. 14, no. 2, pp. 345–365, 2012. [Online]. Available: http://www.iwaponline.com/jh/014/jh0140345.htm

[38] M. Gavanelli, M. Nonato, A. Peano, S. Alvisi, and M. Franchini, "Scheduling countermeasures to contamination events by genetic algorithms," *AI Communications*, vol. 28, no. 2, pp. 259–282, 2015. [Online]. Available: http://iospress.metapress.com/content/lr175378659g4r73/

[39] ——, "Genetic algorithms for scheduling devices operation in a water distribution system in response to contamination events," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, J.-K. Hao and M. Middendorf, Eds. Springer Berlin / Heidelberg, 2012, vol. 7245, pp. 124–135. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29124-1_11

[40] S. Liu, H. Che, K. Smith, and L. Chen, "Contamination event detection using multiple types of conventional water quality sensors in source water," *Environ. Sci.: Processes Impacts*, vol. 16, pp. 2028–2038, 2014. [Online]. Available: http://dx.doi.org/10.1039/C4EM00188E

[41] 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*, vol. 134, no. 6, pp. 516–526, 2008. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(2008)134:6(516)

[42] T. M. Baranowski and E. J. LeBoeuf, "Consequence management optimization for contaminant detection and isolation," *Journal of Water Resources Planning and Management*, vol. 132, no. 4, pp. 274–282, 2006. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(2006)132:4(274)

[43] M. E. Shafiee and E. Z. Berglund, "Real-time guidance for hydrant flushing using sensor-hydrant decision trees," *Journal of Water Resources Planning and Management*, vol. 0, no. 0, p. 0, 2014, accepted for publication. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000475

[44] 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*, vol. 124, no. 2, pp. 267 – 282, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722179900380X

[45] C. J. Malmborg, "A genetic algorithm for service level based vehicle scheduling," *European Journal of Operational Research*, vol. 93, no. 1, pp. 121 – 134, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221795001859

[46] 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*, vol. 175, no. 1, pp. 246 – 257, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221705004236

[47] ——, "Scheduling pre-printed newspaper advertising inserts using genetic algorithms," *Omega*, vol. 30, no. 6, pp. 415 – 421, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048302000592

[48] 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 (TAAI), 2011 International Conference on*, Nov 2011, pp. 331–336. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6120767

[49] G. Reinelt, "TSPLIB - A t.s.p. library," Universität Augsburg, Institut für Mathematik, Augsburg, Tech. Rep. 250, 1990. [Online]. Available: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[50] 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*, vol. 10, no. 3, pp. 232 – 243, 2003.

[51] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, no. 1, pp. 77 – 95, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221704002656

[52] F. Sivrikaya-Şerifoğlu and G. Ulusoy, "Parallel machine scheduling with earliness and tardiness penalties," *Computers & Operations Research*, vol. 26, no. 8, pp. 773 – 787, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054898000902

[53] T. Zhang, W. Gruver, and M. Smith, "Team scheduling by genetic search," in *Intelligent Processing and Manufacturing of Materials, 1999. IPMM '99. Proceedings of the Second International Conference on*, vol. 2, 1999, pp. 839–844 vol.2. [Online]. Available: http://dx.doi.org/10.1109/IPMM.1999.791495

[54] R. Cheng and M. Gen, "Parallel machine scheduling problems using memetic algorithms," *Computers & Industrial Engineering*, vol. 33, no. 34, pp. 761 – 764, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835297002477

[55] P. A. Rubin and G. L. Ragatz, "Scheduling in a sequence dependent setup environment with genetic search," *Computers & Operations Research*, vol. 22, no. 1, pp. 85 – 99, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0305054893E0021K

[56] 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*, vol. 187, no. 3, pp. 985 – 1032, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706008174

[57] R. G. Regis and C. A. Shoemaker, “Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization,” *Engineering Optimization*, vol. 45, no. 5, pp. 529–555, 2013. [Online]. Available: http://dx.doi.org/10.1080/0305215X.2012.687731

[58] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[59] P. Toth and D. Vigo, *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds. Society for Industrial and Applied Mathematics, 2002. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9780898718515

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

[61] Gurobi Optimization, Inc., “Gurobi optimizer reference manual,” 2014, http://www.gurobi.com.

[62] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522

[63] E. C. Myles Hollander, Douglas A. Wolfe, *Nonparametric statistical methods*, 3rd ed., ser. Wiley series in probability and statistics: Texts and references section. Wiley, 2014. [Online]. Available: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470387378.html

[64] P. Nemenyi, “Distribution-free multiple comparisons,” Ph.D. dissertation, Princeton University, 1963.

[65] C. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilità*, ser. Pubblicazioni del R. Istituto superiore di scienze economiche e commerciali di Firenze. Libreria internazionale Seeber, 1936.

[66] R. Miller, *Simultaneous Statistical Inference*, ser. Springer Series in Statistics. Springer New York, 1981. [Online]. Available: http://www.springer.com/us/book/9781461381242

[67] Addinsoft SARL, “XLSTAT v2012.6,” http://www.xlstat.com.

[68] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and Cybernetics*, vol. 39, pp. 653–684, 2000. [Online]. Available: http://leeds-faculty.colorado.edu/glover/ssandprfundamentals.pdf

[69] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan, "Rich vehicle routing problem: Survey," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 32:1–32:28, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2666003

[70] S. Ho and M. Gendreau, "Path relinking for the vehicle routing problem," *Journal of Heuristics*, vol. 12, no. 1-2, pp. 55–72, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10732-006-4192-1

[71] C. Prins, C. Prodhon, and R. Calvo, "Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking," *4OR*, vol. 4, no. 3, pp. 221–238, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10288-006-0001-9

[72] M. Reghioui, C. Prins, and N. Labadi, "Grasp with path relinking for the capacitated arc routing problem with time windows," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, M. Giacobini, Ed. Springer Berlin Heidelberg, 2007, vol. 4448, pp. 722–731. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71805-5_78

[73] K. Srensen and P. Schittekat, "Statistical analysis of distance-based path relinking for the capacitated vehicle routing problem," *Computers & Operations Research*, vol. 40, no. 12, pp. 3197 – 3205, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054813000373

[74] A. Rahimi-Vahed, T. Crainic, M. Gendreau, and W. Rei, "A path relinking algorithm for a multi-depot periodic vehicle routing problem," *Journal of Heuristics*, vol. 19, no. 3, pp. 497–524, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10732-013-9221-2

[75] E. Vallada and R. Ruiz, "Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem," *Omega*, vol. 38, no. 12, pp. 57 – 67, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048309000322

[76] G. Q. Zhang and K. K. Lai, "Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem," *European*

Journal of Operational Research, vol. 169, no. 2, pp. 413 – 425, 2006, feature Cluster on Scatter Search Methods for Optimization. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221704005466

[77] M. Ranjbar, F. Kianfar, and S. Shadrokh, "Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm," *Applied Mathematics and Computation*, vol. 196, no. 2, pp. 879 – 888, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300307007412

[78] M. Martins, S. Fuchs, L. Pando, R. Lders, and M. Delgado, "PSO with path relinking for resource allocation using simulation optimization," *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 322 – 330, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835213000417

[79] A. Peano, M. Nonato, M. Gavanelli, S. Alvisi, and M. Franchini, "A bilevel mixed integer linear programming model for valves location in water distribution systems," in *3rd Student Conference on Operational Research*, ser. OpenAccess Series in Informatics (OASIcs), S. Ravizza and P. Holborn, Eds., vol. 22. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 103–112. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2012/3551

[80] M. Gavanelli, M. Nonato, A. Peano, S. Alvisi, and M. Franchini, "An ASP approach for the valves positioning optimization in a water distribution system," in *9th Italian Convention on Computational Logic (CILC 2012), Rome, Italy*, ser. CEUR workshop proceedings, F. Lisi, Ed., vol. 857, 2012, pp. 134–148. [Online]. Available: http://ceur-ws.org/Vol-857/paper_f10.pdf

[81] M. Gavanelli, M. Nonato, and A. Peano, "An ASP approach for the valves positioning optimization in a water distribution system," *Journal of Logic and Computation*, 2013, accepted for publication. [Online]. Available: http://logcom.oxfordjournals.org/content/early/2013/12/04/logcom.ext065

[82] H. Jun and G. V. Loganathan, "Valve-controlled segments in water distribution systems," *Journal of Water Resources Planning and Management*, vol. 133, no. 2, pp. 145–155, 2007. [Online]. Available: http://dx.doi.org/10.1061/(ASCE)0733-9496(2007)133:2(145)

[83] J.-J. Kao and P.-H. Li, "A segment-based optimization model for water pipeline replacement," *Journal - American Water*

Works Association, vol. 99, no. 7, pp. 83–95, 2007. [Online]. Available: http://www.awwa.org/publications/journal-awwa/abstract/articleid/15698

[84] M. Bruni, P. Beraldi, and D. Conforti, "A stochastic programming approach for the strategic valve locations problem in a water distribution system," *Procedia - Social and Behavioral Sciences*, vol. 108, no. 0, pp. 129 – 138, 2014, operational Research for Development, Sustainability and Local Economies. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042813054669

[85] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical Journal*, vol. 49, pp. 291–307, 1970. [Online]. Available: http://www.cs.bell-labs.com/who/bwk/partitioning.pdf

[86] F. R. Chung, *Spectral graph theory.* American Mathematical Soc., 1997, vol. 92.

[87] R. Pichler, S. Rümmele, and S. Woltran, "Multicut algorithms via tree decompositions," in *Algorithms and Complexity*, ser. Lecture Notes in Computer Science, T. Calamoneri and J. Diaz, Eds., vol. 6078. Springer Berlin Heidelberg, 2010, pp. 167–179. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13073-1_16

[88] S. Chopra and M. R. Rao, "The partition problem," *Mathematical Programming*, vol. 59, no. 1-3, pp. 87–115, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF01581239

[89] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, and L. Wolsey, "Formulations and valid inequalities for the node capacitated graph partitioning problem," *Mathematical Programming*, vol. 74, no. 3, pp. 247–266, 1996. [Online]. Available: http://dx.doi.org/10.1007/BF02592198

[90] A. D. Nardo, M. D. Natale, G. Santonastaso, and S. Venticinque, "Graph partitioning for automatic sectorization of a water distribution system," in *Urban Water Management: Challenges and Opportunities*, D. Savic, Z. Kapelan, and D. Butler, Eds., vol. 3. Centre for Water Systems, University of Exeter, Exeter (UK), 2011, pp. 841–846.

[91] "IBM ILOG CPLEX Optimizer, user's manual for CPLEX," 2010. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[92] T. Achterberg, "Scip: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, July 2009. [Online]. Available: http://mpc.zib.de/index.php/MPC/article/view/4

[93] Fair Isaac Corporation, "FICO Xpress Optimization Suite." [Online]. Available: http://www.fico.com/en/products/fico-xpress-optimization-suite

[94] B. Gendron, T. Crainic, and A. Frangioni, "Multicommodity capacitated network design," in *Telecommunications Network Planning*, ser. Centre for Research on Transportation, B. Sans and P. Soriano, Eds. Springer US, 1999, pp. 1–19. [Online]. Available: http://dx.doi.org/10.1007/978-1-4615-5087-7_1

[95] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol. 153, no. 1, pp. 235–256, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10479-007-0176-2

[96] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, ser. Dover Books on Computer Science Series. Dover Publications, 1998. [Online]. Available: https://books.google.it/books?id=cDY-joeCGoIC

[97] V. Kaibel, M. Peinhardt, and M. E. Pfetsch, "Orbitopal fixing," *Discrete Optimization*, vol. 8, no. 4, pp. 595 – 610, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1572528611000430

[98] M. Bruglieri, P. Cappanera, A. Colorni, and M. Nonato, "Modeling the gateway location problem for multicommodity flow rerouting," in *Network Optimization*, ser. Lecture Notes in Computer Science, J. Pahl, T. Reiners, and S. Vo, Eds. Springer Berlin Heidelberg, 2011, vol. 6701, pp. 262–276. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21527-8_31

[99] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, ser. Always learning. Pearson, 2014. [Online]. Available: https://books.google.it/books?id=DFJtngEACAAJ

[100] W. Clocksin and C. Mellish, *Programming in Prolog*. Springer Berlin Heidelberg, 2003. [Online]. Available: https://books.google.it/books?id=VjHk2Cjrti8C

[101] N.-f. Zhou, "The language features and architecture of b-prolog," *Theory Pract. Log. Program.*, vol. 12, no. 1-2, pp. 189–218, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1017/S1471068411000445

[102] K. R. Apt and M. Wallace, *Constraint Logic Programming using ECLiPSe.* Cambridge University Press, 2007.

[103] M. Carlsson and P. Mildner, "Sicstus prologthe first 25 years," *Theory and Practice of Logic Programming*, vol. 12, pp. 35–66, 1 2012. [Online]. Available: http://journals.cambridge.org/article_S1471068411000482

[104] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager, "Swi-prolog," *Theory and Practice of Logic Programming*, vol. 12, pp. 67–96, 1 2012. [Online]. Available: http://journals.cambridge.org/article_S1471068411000494

[105] C. Baral, *Knowledge Representation, Reasoning, and Declarative Problem Solving.* New York, NY, USA: Cambridge University Press, 2003.

[106] N. Leone, "Logic programming and nonmonotonic reasoning: From theory to systems and applications," in *Logic Programming and Nonmonotonic Reasoning*, ser. Lecture Notes in Computer Science, C. Baral, G. Brewka, and J. Schlipf, Eds. Springer Berlin Heidelberg, 2007, vol. 4483, pp. 1–1. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72200-7_1

[107] E. Giunchiglia, Y. Lierler, and M. Maratea, "Answer set programming based on propositional satisfiability," *Journal of Automated Reasoning*, vol. 36, pp. 345–377, 2006. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/?giu06

[108] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello, "The DLV system for knowledge representation and reasoning," *ACM Transactions on Computational Logic (TOCL)*, vol. 7, no. 3, pp. 499–562, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1149114.1149117

[109] M. Gebser, B. Kaufmann, and T. Schaub, "Conflict-driven answer set solving: From theory to practice," *Artificial Intelligence*, vol. 187-188, pp. 52–89, Aug. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2012.04.001

[110] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proceedings of International Logic Programming Conference and Symposium*, R. Kowalski, Bowen, and Kenneth, Eds. MIT Press, 1988, pp. 1070–1080. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/?gel88

[111] M. Gebser, R. Kaminski, M. Ostrowski, T. Schaub, and S. Thiele, “On the input language of asp grounder gringo,” in *Logic Programming and Nonmonotonic Reasoning*, ser. Lecture Notes in Computer Science, E. Erdem, F. Lin, and T. Schaub, Eds., vol. 5753. Springer Berlin Heidelberg, 2009, pp. 502–508. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04238-6_49

[112] P. Simons, I. Niemelä, and T. Soininen, “Extending and implementing the stable model semantics,” *Artificial Intelligence*, vol. 138, no. 12, pp. 181 – 234, 2002, knowledge Representation and Logic Programming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437020200187X

[113] F. Lin and Y. Zhao, “ASSAT: computing answer sets of a logic program by SAT solvers,” *Artificial Intelligence*, vol. 157, no. 12, pp. 115 – 137, 2004, nonmonotonic Reasoning. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370204000578

[114] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider, “Potassco: The potsdam answer set solving collection,” *AI Communications*, vol. 24, no. 2, pp. 107–124, Apr. 2011. [Online]. Available: http://iospress.metapress.com/content/nr20503mu5606v0x/

[115] W. Faber, N. Leone, and G. Pfeifer, “Recursive aggregates in disjunctive logic programs: Semantics and complexity,” in *Logics in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Alferes and J. Leite, Eds., vol. 3229. Springer Berlin Heidelberg, 2004, pp. 200–212. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30227-8_19

[116] O. Goldschmidt and D. S. Hochbaum, “A polynomial algorithm for the k-cut problem for fixed k,” *Mathematics of Operations Research*, vol. 19, no. 1, pp. 24–37, 1994. [Online]. Available: http://dx.doi.org/10.1287/moor.19.1.24

[117] J. D. Horton, “A polynomial-time algorithm to find the shortest cycle basis of a graph,” *SIAM Journal on Computing*, vol. 16, no. 2, pp. 358–366, 1987. [Online]. Available: http://dx.doi.org/10.1137/0216026

[118] K. Mehlhorn and D. Michail, “Implementing minimum cycle basis algorithms,” *Journal of Experimental Algorithmics*, vol. 11, Feb. 2007. [Online]. Available: http://doi.acm.org/10.1145/1187436.1216582

[119] F. Berger, P. Gritzmann, and S. de Vries, "Minimum cycle bases for network graphs," *Algorithmica*, vol. 40, no. 1, pp. 51–62, 2004. [Online]. Available: http://dx.doi.org/10.1007/s00453-004-1098-x

[120] J. C. de Pina, "Applications of shortest path methods," Ph.D. dissertation, Amsterdam School of Economics Research Institute (ASE-RI), 1995.

[121] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM*, vol. 37, no. 2, pp. 318–334, 1990. [Online]. Available: http://doi.acm.org/10.1145/77600.77620

[122] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems." *Numerische Mathematik*, vol. 4, pp. 238–252, 1962/63. [Online]. Available: http://eudml.org/doc/131533

[123] A. Geoffrion, "Generalized benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, 1972. [Online]. Available: http://dx.doi.org/10.1007/BF00934810

[124] J. Hooker, *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011. [Online]. Available: https://books.google.it/books?id=1fT6qiih5ygC

[125] A. M. Costa, "A survey on benders decomposition applied to fixed-charge network design problems," *Computers & Operations Research*, vol. 32, no. 6, pp. 1429 – 1450, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054803003435

Author’s Publications List

[J1] Marco Gavanelli, Maddalena Nonato, Andrea Peano, Stefano Alvisi, and Marco Franchini, “Scheduling countermeasures to contamination events by genetic algorithms”, *AI Communications, vol. 28, no. 2, pp. 259–282, 2015*
doi: 10.3233/AIC-140638.
Scopus: 2-s2.0-84922560199.
ISI: 000349156700007.

[J2] Marco Gavanelli, Maddalena Nonato, and Andrea Peano, “An ASP approach for the valves positioning optimization in a water distribution system”, *Journal of Logic and Computation, accepted for publication.*
doi: 10.1093/logcom/ext065.

[C1] Marco Gavanelli, Maddalena Nonato, Andrea Peano, Stefano Alvisi, and Marco 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*
doi: 10.1007/978-3-642-29124-1_11
Scopus: 2-s2.0-84859138145.

[C2] Andrea Peano, Maddalena Nonato, Marco Gavanelli, Stefano Alvisi, and Marco Franchini, “A bilevel mixed integer linear programming model for valves location in water distribution systems”, *In S. Ravizza and P. L. Holborn, editors, SCOR, volume 22 of OASICS, pages 103–112. Schloss Dagstuhl- Leibniz-Zentrum fuer Informatik, 2012*
doi: 10.4230/OASIcs.SCOR.2012.103.

[W1] Andrea Peano, "An ASP approach for the optimal placement of the isolation valves in a water distribution system", *In A. Dovier and V. S. Costa, editors, Technical Communications of the 28th International Conference on Logic Programming (ICLP12), volume 17 of Leibniz International Proceedings in Informatics (LIPIcs), pages 464–468, Dagstuhl, Germany, 2012. Schloss DagstuhlLeibniz-Zentrum fuer Informatik*
doi: 10.4230/LIPIcs.ICLP.2012.464
Scopus: 2-s2.0-84880191828.

[W2] Andrea Peano, Marco Gavanelli, "An ASP approach for the optimal placement of the isolation valves in a water distribution system", *In P. Liberatore, editor, Doctoral Consortium of the 12th Symposium of the Italian Association for Artificial Intelligence (AIxIA-DC 2012) , volume 926 of CEUR Workshop Proceedings, pages: 33–37, 2012*
Scopus: 2-s2.0-84891763101.

[W3] Marco Gavanelli, Maddalena Nonato, Andrea Peano, Stefano Alvisi, and Marco Franchini, "An ASP approach for the valves positioning optimization in a water distribution system", *In F. Lisi, editor, 9th Italian Convention on Computational Logic (CILC 2012), Rome, Italy, volume 857 of CEUR workshop proceedings, pages 134–148, 2012*
Scopus: 2-s2.0-84883383155.