Optim Eng (2012) 13:219–246
DOI 10.1007/s11081-011-9141-7

On the optimal design of water distribution networks: a practical MINLP approach

Cristiana Bragalli · Claudia D'Ambrosio · Jon Lee · Andrea Lodi · Paolo Toth

Received: 18 September 2009 / Accepted: 3 March 2011 / Published online: 23 March 2011

Abstract We propose a practical solution method for real-world instances of a water-network optimization problem with fixed topology using a nonconvex continuous NLP (NonLinear Programming) relaxation and a MINLP (Mixed Integer NonLinear Programming) search. Our approach employs a relatively simple and accurate model that pays some attention to the requirements of the solvers that we employ. Our view is that in doing so, with the goal of calculating only good feasible solutions, complicated algorithmics can be confined to the MINLP solver. We report successful computational experience using available open-source MINLP software on problems from the literature and on difficult real-world instances. An important contribution of this paper is that the solutions obtained, besides being low cost, are immediately usable in practice because they are characterized by an allocation of diameters to pipes that leads to a correct hydraulic operation of the network. This is not the case for most of the other methods presented in the literature.

Keywords Water network design · Mixed-integer nonlinear programming · Modeling · Computation

C. Bragalli
DISTART, University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy
e-mail: cristiana.bragalli@mail.ing.unibo.it

C. D'Ambrosio · A. Lodi (✉) · P. Toth
DEIS, University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy
e-mail: andrea.lodi@unibo.it

C. D'Ambrosio
e-mail: c.dambrosio@unibo.it

P. Toth
e-mail: paolo.toth@unibo.it

J. Lee
IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, USA
e-mail: jonlee@us.ibm.com

1 Introduction

The optimal design of a WDN (Water Distribution Network) consists, in its classical formulation, of the choice of a diameter for each pipe, while other design properties are considered to be fixed (e.g., the topology and pipe lengths). From a mathematical viewpoint, we can cast the optimal design problem of a WDN as an MINLP (Mixed Integer NonLinear Programming) problem in which the discrete variables select from a set of commercially-available diameters, water flows and pressures must respect the hydraulic constraints, and we seek to minimize the cost function which only depends on the selected diameters. Note that no pumping operation to raise the head of the reservoirs is considered in this paper.

Recently there has been renewed interest in optimal WDN design, due to emerging issues related to water distribution systems; in particular, the gradual deterioration of network pipes and the need for a more rational use of water resources has led to very costly renovation activities.

Approaches in the literature use various combinations of linearization and relaxation, which lead to MILP (Mixed Integer Linear Programming), NLP (NonLinear Programming) and meta-heuristic algorithms. We survey these approaches in Sect. 4. In this paper we are interested in approaches exploiting mathematical-programming formulations, and we consider two cases.

The MILP approach to our problem relies on using piecewise-linear approximations. If tractable, a solution of such a model would provide a global optimum of an approximation to the real system. If accurate models are desired for a large network, we are led to using a large number of binary variables (to manage the linear pieces). This tends to lead to a very poor relaxation and ultimately an intractable model.

With an MINLP approach, we are led to a more natural model. Our view is that by accurately modeling the nonlinear phenomena, we will have a model that will provide an MINLP search with a good NLP relaxation. While foregoing any hope of practically verifying MINLP global optimality of the best solution obtained, we are able to find very good solutions to large real-world instances.

Our experiments were carried out using AMPL (Fourer et al. 2003) as an interface to MINLP codes. In a preliminary version of this work Bragalli et al. (2006), we used Leyffer's code `MINLP_BB` (Leyffer 1998, available from the University of Dundee) as well as the—at that time new—CMU/IBM open-source MINLP code `Bonmin` (Bonami et al. 2008; Bonami and Lee 2006; Bonmin v. 0.1), available from COIN-OR. In fact, it was in the context of our investigations that `Bonmin` was adapted, with the introduction of new features, for use on nonconvex MINLP problems.

Our modeling and solution methods were worked out with the target software in mind (in particular, the branch-and-bound implementation in Bonmin (v. 0.1)), our results were all obtained by implementing our special features within the development (trunk) version of `Bonmin`. We note that the open-source nature of `Bonmin` enabled us to rapidly test our ideas and then make them available to the developers and the users of `Bonmin` under the same open-source license used by `Bonmin` (Common Public License Version 1.0 (CPL)).

Main contributions. The use of MINLP techniques and software for the *practical solution* of optimal design of water distribution networks is the main contribution of the present paper.

On the one side, mathematical programming techniques such as MILP and NLP have been already used in this context, but previous such efforts considered relaxations of the problem or dealt with *small-scale* instances (see, Sect. 2 and Sect. 4.1 for details). On the other hand, meta-heuristic approaches considered *medium-* to *large-scale* real-world instances of WDN, but of course without the flexibility and accuracy of the mathematical modeling.

Due to very recent advances in MINLP software, we address WDN with MINLP techniques which simultaneously allow us to handle the nonlinear and discrete parts of the model accurately and provide an effective and reliable framework to perform extensive, practical computation.

A second methodological contribution of the paper is on the development of techniques to handle the nonconvexity of the model. The effectiveness of these techniques is shown by the fact that, starting from the development version, they have now been incorporated in the latest release of the successful open-source MINLP software `Bonmin` (Bonmin v. 1.0).

Finally, an important characteristic of the solutions obtained is that, besides being low cost, they are immediately usable in practice. The reason for that is twofold. On the one hand, we explicitly take into account the operational constraints on the speed of the water through the pipes. On the other hand, our solutions are naturally characterized by an allocation of diameters to pipes that leads to a correct hydraulic operation of the network. This last issue is discussed in details in Sect. 6.4, and it is yet another important difference compared to meta-heuristic approaches whose solutions require non-trivial postprocessing before they can be used in practice.

In Sect. 2, we formally set notation for specifying instances of the problem. In Sect. 3, we describe the problem more fully, through a preliminary continuous model and we discuss the two main modeling contributions of the paper, namely a continuous objective function (see, Sect. 3.1) and a smooth (approximate) relaxation of the pressure loss in water pipes (see, Sect. 3.2). In Sect. 4.1, we survey earlier approaches, while in Sect. 4.2 we describe how we incorporate binary variables for the purposes of then applying MINLP codes. In Sect. 4.3, so as to decrease the nonlinearity and nonconvexity, we describe a reparameterization of pipes dimension by (cross-sectional) area, rather than diameter. In Sect. 5, we describe the results of computational experiments and in Sect. 6 we evaluate these results with respect to different metrics and characteristics. Finally, in Sect. 7 we draw some conclusions.

2 Notation

The network is oriented for the sake of making a careful formulation, but flow on each pipe is not constrained in sign (i.e., it can be in either direction). The network consists of pipes (arcs) and junctions (nodes). In the optimization, the pipes are to have their diameters sized at minimum cost.

Sets:

E = set of pipes.
N = set of junctions.

S = set of source junctions (also called reservoirs, $S \subset N$).
$\delta_+(i)$ = set of pipes with tail at junction i ($i \in N$).
$\delta_-(i)$ = set of pipes with head at junction i ($i \in N$).

Parameters:

$len(e)$ = length of pipe e ($e \in E$).
$k(e)$ = physical constant depending on the roughness of pipe e ($e \in E$).
$d_{min}(e)$ = minimum diameter of pipe e ($e \in E$).
$d_{max}(e)$ = maximum diameter of pipe e ($e \in E$).
$v_{max}(e)$ = maximum speed of water in pipe e ($e \in E$).
$dem(i)$ = demand at junction i ($i \in N \setminus S$).
$elev(i)$ = physical elevation of junction i ($i \in N \setminus S$).
$ph_{min}(i)$ = minimum pressure head at junction i ($i \in N \setminus S$).
$ph_{max}(i)$ = maximum pressure head at junction i ($i \in N \setminus S$).
$h_s(i)$ = fixed hydraulic head of source junction i ($i \in S$).

For each pipe e, the available diameters belong to a discrete set of r_e elements. For $e \in E$:

$$d_{min}(e) := \mathfrak{D}(e, 1) < \mathfrak{D}(e, 2) < \cdots < \mathfrak{D}(e, r_e) =: d_{max}(e).$$

For each pipe $e \in E$, there is a cost function $C_e()$ having a discrete specification as a (typically rapidly) increasing function of diameter. That is, $\mathfrak{C}(e, r) := C_e(\mathfrak{D}(e, r))$, $r = 1, \ldots, r_e$, where:

$$\mathfrak{C}(e, 1) < \mathfrak{C}(e, 2) < \cdots < \mathfrak{C}(e, r_e).$$

As mentioned in Sect. 1, different techniques to attack this problem have been proposed in the literature. In particular, an interesting approach employs a so-called "split-pipe model": each pipe e is split into r_e stretches of unknown length, where r_e is the number of possible choices of the diameter of pipe e, and variables model the lengths of the stretches. It is not difficult to see that models of this type have the disadvantage of allowing solutions with several changes in the diameter along the length of a pipe, i.e., the considered problem is a relaxation of the original one. Optimality considerations lead to the conclusion that the diameters selected for each pipe can be at most two (see, e.g., Fujiwara and Khang 1990). However, for very large instances even splitting pipes in (only) two pieces can deteriorate the performance of the network because of sometimes significant pressure loss at the junction of the stretches (so-called "minor head losses"). Such losses are ignored by all of the optimization models we are aware of.

3 A preliminary continuous model

In this section, we describe the problem, and at the same time we develop a preliminary NLP relaxation. Our goal is to develop a smooth NLP formulation that accurately models the problem.

Variables:

$Q(e)$ = flow in pipe e ($\forall e \in E$).
$D(e)$ = diameter of pipe e ($\forall e \in E$).
$H(i)$ = hydraulic head of junction i ($\forall i \in N$).

Simple bounds [Linear]:

$d_{min}(e) \leq D(e) \leq d_{max}(e)$ ($\forall e \in E$).
$ph_{min}(i) + elev(i) \leq H(i) \leq ph_{max}(i) + elev(i)$ ($\forall i \in N \setminus S$).
$H(i) = h_s(i)$ ($\forall i \in S$).

The hydraulic head is the total energy per unit of weight of the water, and it is expressed in terms of a height. Furthermore, the hydraulic head is the sum of pressure head (ph), elevation head ($elev$) and velocity head ($\frac{v^2}{2g}$), all of which are measured in units of length. Velocity head (kinetic energy) is usually ignored because is much smaller than the elevation and pressure head (see Walski et al. 2001).

Flow bounds (dependent on cross-sectional area of pipe) [Smooth but nonconvex]:

$$-\frac{\pi}{4} v_{max}(e) D^2(e) \leq Q(e) \leq \frac{\pi}{4} v_{max}(e) D^2(e) \quad (\forall e \in E).$$

The formulation also considers the important design criteria relating to the maximum velocity (v_{max}) of the water in the pipes that must not exceed an appropriate value. Because the velocity is not an explicit variable of the model, this bound is expressed as a function of the flow by using the relationship between flow, velocity and diameter.

Flow conservation [Linear]:

$$\sum_{e \in \delta_-(i)} Q(e) - \sum_{e \in \delta_+(i)} Q(e) = dem(i) \quad (\forall i \in N \setminus S).$$

For each node, the flow conservation constraints insure that the difference between the sum of the pipe flows entering the node and the sum of the pipe flows exiting it is equal to the water demand at the node (assumed positive when water is consumed).

Head loss across links [Nonsmooth and nonconvex]:

$$H(i) - H(j) = \text{sgn}(Q(e)) |Q(e)|^{1.852} \cdot 10.7 \cdot len(e) \cdot k(e)^{-1.852} / D(e)^{4.87}$$
$$(\forall e = (i, j) \in E).$$

This last constraint models pressure loss in water pipes due to friction using the empirical *Hazen-Williams equation*. For each pipe e, the equation uses a single constant $k(e)$ to characterize the roughness of the pipes inner surface which only depends on the material the pipe is made. Introduced in 1902, the Hazen-Williams equation is

an accepted model for fully turbulent flow in water networks (see Walski 1984) that, because of its simplicity, has had large diffusion in hydraulic computations.

Diameter is bounded away from 0, so the only nondifferentiability is when the flow is 0. Such a nondifferentiability is discussed in details in Sect. 3.2.

Objective to be minimized [Discrete]:

$$\sum_{e \in E} C_e(D(e))\ len(e).$$

Because we only have discretized cost data, within AMPL we are fitting a polynomial to the input discrete cost data to make a *smooth* working continuous cost function $C_e()$.

Our motivation for that is to use a smooth function to closely fit the discrete cost data and the details of such a choice together with the relationship between continuous and discrete objective functions are discussed in the following section. In addition, computational experiments comparing the two options are reported at the end of Sect. 5.2.2.

3.1 Objective function

We have experimented with different fits: l_1, l_2 and l_∞; with and without requiring that the fit under or over approximates the discrete points. Requiring an under approximation makes our formulation a true relaxation—in the sense that the global minimum of our relaxation is a lower bound on the discrete optimum. We use and advocate weighted fits to minimize relative error. For example, our least-squares fit for pipe e minimizes

$$\sum_{r=1}^{r_e} \frac{1}{\mathfrak{C}(e,r)^2} \left[\mathfrak{C}(e,r) - \left(\sum_{j=0}^{t_e} \beta(j,e) \left(\frac{\pi}{4} \mathfrak{D}(e,r)^2 \right)^j \right) \right]^2$$

$$= \sum_{r=1}^{r_e} \left[1 - \left(\frac{\sum_{j=0}^{t_e} \beta(j,e) (\frac{\pi}{4} \mathfrak{D}(e,r)^2)^j}{\mathfrak{C}(e,r)} \right) \right]^2,$$

where t_e is the desired degree and $\beta(j,e)$ are the coefficients of the polynomial[1] approximating C_e.

We have experimented with several low-degree polynomials in order to find a satisfactory approximation. Note that for each pipe of this instance, the set of diameters is the same, thus we used the same continuous cost function for each pipe. The polynomial that best fits our purposes especially for these important diameters is the one of degree 7. Note that, we do not insure that the polynomial is increasing nor convex,

[1] Note that the least-square minimization is itself a nonconvex NLP that we solve to local optimality using the open-source NLP solver `Ipopt` (Ipopt v. 3.5) which we use as an NLP solver throughout our work (see Sect. 5.2).

and actually we do not even assume this for the data, though for the data sets that we experimented with the discrete data are increasing.

We will come back to the choice of discrete vs continuous objective function in Sect. 4.2 and we will report some computational experiments in Sect. 5.2.2.

Before ending the section we note that one drawback to using a low-degree polynomial (for each pipe) to fit the discrete costs is that this would attain the correct value of the objective function for each integer solution only if there is a low-degree polynomial that has a relative error equal to 0. As this is unlikely, we may have to make a compromise, in modeling the objective function, between modeling accuracy and numerical behavior.

This difficulty can be overcome in an alternative manner. We can instead define a continuous objective function so as to fit the discrete values $\mathfrak{C}(e, r)$ using a cubic spline for each pipe e. Each piece of a cubic spline is a degree-three polynomial that passes between a pair of consecutive discrete points $(\mathfrak{D}(e, r-1), \mathfrak{C}(e, r-1))$ and $(\mathfrak{D}(e, r), \mathfrak{C}(e, r))$ $(e \in E,\ r = 2, \ldots, r_e)$. The use of cubic splines guarantees that, once an integer solution is found, its objective value is correct.

This piecewise definition of the function can be easily accommodated using a modeling language like AMPL (which has a natural syntax for defining piecewise functions). However, the NLP solvers, and in particular `Ipopt` (see Sect. 5.2), seem to more easily manage a polynomial with high degree, as compared to $r_e - 1$ different polynomials pieces of degree 3, thus the experiments in Sect. 5 use the single polynomial objective function with an algorithmic correction for taking into account the original discrete one (see Sect. 5.2). We do note, however, that we believe that the spline approach has considerable potential, but more work would be needed on the side of NLP solvers to realize a computational benefit.

3.2 Smoothing the nondifferentiability

The main remaining modeling difficulty is to deal algorithmically with the absolute value term in the head loss constraints. This term is nondifferentiable (at 0) but not badly so. One possibility is to ignore the nondifferentiability issue, and just use a solver that will handle it in its own way. This has the advantage of straightforward implementation from AMPL and access to many NLP solvers (e.g., via NEOS NEOS v. 5.0). Because our ultimate goal is, however, to employ available MINLP solvers, we tested such a straightforward approach by using the MINLP solver `Bonmin` and its default NLP solver `Ipopt` (see Sect. 5.2). As expected, `Ipopt` is unable to handle such a nondifferentiable function, and it aborts the run immediately.

Thus, to accommodate the NLP solver, we had to smooth the nondifferentiability, and in order to do that effectively, our main goal is not to provide a fully accurate approximation near 0 because it is well known that Hazen-Williams equation is in itself a poor approximation of the real pressure loss for small values of the flow. Instead, we smooth away the mild nondifferentiability by defining the head loss equation in a piecewise manner, in such a manner as to have accurate evaluations of the function. We insure the smoothness by matching function values as well as first and second derivative at the breakpoints.

More precisely, let $f(x) = x^p$ $(p = 1.852)$ when x is nonnegative, and $f(x) = -f(-x)$ when x is negative (x is standing in for $Q(e)$). This function misbehaves at

0 (the second derivative does not exist there). Choose a small positive δ, and replace f with a function g on $[-\delta, +\delta]$. Outside of the interval, we leave f alone. We will choose g to be of the following form: $g(x) = ax + bx^3 + cx^5$. In this way, we can choose a, b, c (uniquely) so that f and g agree in value, derivative and second derivative, at $x = |\delta|$. So we end up with a smooth-enough anti-symmetric function. It agrees in value with f at 0 and outside $[-\delta, +\delta]$. It agrees with f in the first two derivatives outside of $[-\delta, +\delta]$.

Formally, it is easy to prove that:

Proposition 1 *The unique polynomial* $g(x) = ax + bx^3 + cx^5$ *having* $f(x) = g(x)$, $f'(x) = g'(x)$ *and* $f''(x) = g''(x)$ *at* $x = |\delta|$ *is*:

$$
\begin{aligned}
g(x) = &\left(\frac{3\delta^{p-5}}{8} + \frac{1}{8}(p-1)p\delta^{p-5} - \frac{3}{8}p\delta^{p-5}\right)x^5 \\
&+ \left(-\frac{5\delta^{p-3}}{4} - \frac{1}{4}(p-1)p\delta^{p-3} + \frac{5}{4}p\delta^{p-3}\right)x^3 \\
&+ \left(\frac{15\delta^{p-1}}{8} + \frac{1}{8}(p-1)p\delta^{p-1} - \frac{7}{8}p\delta^{p-1}\right)x.
\end{aligned}
$$

Proof Via simple calculation one simply has to equate: (i) $g(\delta) = a\delta + b\delta^3 + c\delta^5 = \delta^p = f(\delta)$, (ii) $g'(\delta) = a + 3b\delta^2 + 5c\delta^4 = p\delta^{p-1} = f'(\delta)$, and (iii) $g''(\delta) = 6b\delta + 20c\delta^3 = p(p-1)\delta^{p-2} = f''(\delta)$. This is now a square linear system in the a, b, c variables. We solve it (symbolically), using *Mathematica* (see Mathematica v. 7.0).

Finally, we just observe that f and g are anti-symmetric, so we have the same a, b, c for $x = -\delta$. □

Figure 1, drawn for $\delta = 0.1$, shows that g provides a good approximation of f. Indeed the quintic curve fits very well on $(-\delta, +\delta)$, and of course it matches up to second order with the true function f at $\pm\delta$. This is all no surprise because we are operating in a small interval of 0, and the function that we approximate is not pathological. The NLP solvers that we have tested appear to respond well to this technique, as does our MINLP solver itself, `Bonmin`.

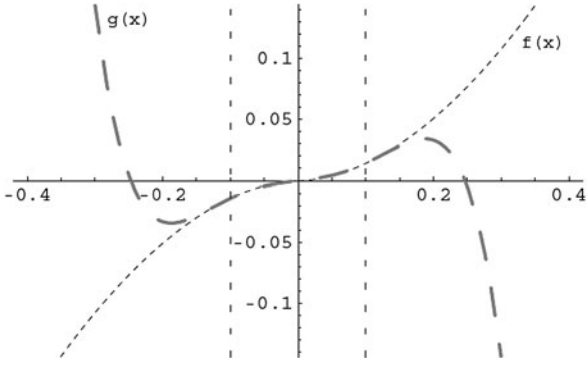

Fig. 1 Smoothing f near $x = 0$

Piecewise constraints can be modeled in AMPL (see §18.3 of Fourer et al. 2003), so we have the advantage of being able to use a variety of NLP solvers, as well as a path to using `Bonmin` and `MINLP_BB`, both of which are interfaced with AMPL. Our experience is that the inaccuracy in using this smoothed function is minimal compared to the other inaccuracies (e.g., numerical and modeling inaccuracies).

4 Models and algorithms

In this section we discuss how to turn our preliminary continuous NLP model into a MINLP that behaves well computationally. For this purpose, we analyze some relevant literature and we then discuss the discrete component of the problem.

4.1 Literature review

Optimal design of a WDN has already received considerable attention. Artina and Walker (1983) linearize and use an MILP approach. Savic and Walters (1997) and Cunha and Sousa (1999) work within an accurate mathematical model, but they use meta-heuristic approaches for the optimization, and they work with the constraints by numerical simulation. Fujiwara and Khang (1990), using the split-pipe model, employ a meta-heuristic approach for the optimization, working with the constraints by numerical simulation. Eiger et al. (1994) also work with a split-pipe model, but they use NLP methods for calculating a solution. Sherali et al. (2001) also work with a split-pipe model, and they successfully employ global optimization methods. Of course, global optimization methods may become impractical for very large scale instances. Lansey and Mays (1989) and Xu and Goulter (1999) also employ an NLP approach, but they use an approximation of the split-pipe methodology (using just two pipe sections). Because the split-pipe model is a relaxation of ours (we only allow a single choice of diameter along the length of a pipe), results using such a model are not directly comparable to ours.

In the rest of the paper, we develop an MINLP approach and compare it to the MILP approach of Artina and Walker (1983). The MILP approach has the advantage of correctly modeling the choices of discrete diameters with binary indicator variables $X(e,r)$ representing the assignment of diameter $\mathfrak{D}(e,r)$ to arc e. In this way we can also easily incorporate costs for the chosen diameters. There is still the nonlinearity of the flow terms in the head loss constraints. Piecewise-linear approximation of these nonlinear constraints is the standard MILP approach here. Unfortunately, the resulting MILPs are typically very difficult to solve. The difficulty of the MILP models is related to the fact that once the diameters have been fixed, the objective function is set, and a feasibility problem associated with the piecewise-linear approximation must be solved, without any guidance from the objective function. It turns out that linear-programming tools in such a context are not effective at all. Good feasible solutions to the models are not always obtainable for even networks of moderate size. Often one is lead to using very coarse piecewise-linear approximations to get some sort of solution, but these tend to not be accurate enough to be considered truly feasible. Indeed, especially with few linearization points, the MILP may (i) generate flows

that are not compatible with the selected diameters because the relation between these variables is only approximated (so the flows computed with the real functions may well be infeasible), and (ii) cut off some feasible (and potentially optimal) solutions. Section 5 includes some of these rather negative computational results obtained with the MILP approach.

4.2 Discretizing the diameters

We need an effective method for imposing the restriction that the diameter of each pipe $e \in E$ belongs to the discrete set of elements:

$$d_{min}(e) := \mathfrak{D}(e, 1) < \mathfrak{D}(e, 2) < \cdots < \mathfrak{D}(e, r_e) =: d_{max}(e).$$

It would be natural and simple to handle this mostly at the level of the MINLP solver, just pass these discrete values to the MINLP solver `Bonmin` via the modeling language (AMPL), and let the MINLP solver construct a two-way branch for a continuous diameter that is strictly between an adjacent pair of discrete choices. Though we could make the necessary changes to the solver `Bonmin`, there does not appear to be a clean way for AMPL to pass such information to the solver. Of course this could be handled in an ad hoc manner, via an auxiliary file, but we prefer to do things in a manner that can be easily and naturally applied to other MINLP solver.

So, for the present, we simply define additional binary variables $X(e, r)$, where $X(e, r) = 1$ indicates that diameter $\mathfrak{D}(e, r)$ is selected for pipe e ($r = 1, \ldots, r_e$, $e \in E$). Then, we use the "SOS type-1" branching (see, Beale and Tomlin 1970) that is available in `Bonmin` (Bonmin v. 1.0). As is standard, we use AMPL suffixes to pass along the SOS information needed by the solver: .sosno ("SOS number") is used to group variables into separate SOS constraints and .ref ("reference value") is used to indicate the value symbolized by a variable. In this way, for $e \in E$, in AMPL we naturally set:

$$X(e, r).sosno := e, \quad \text{for } r = 1, \ldots, r_e,$$

and

$$X(e, r).ref := \mathfrak{D}(e, r), \quad \text{for } r = 1, \ldots, r_e.$$

We note that with the introduction of these binary variables, we could use them in the objective function and eliminate the need for the fitted objective function introduced in Sect. 3. However, to do so would implicitly define a piecewise-linear cost function for each pipe, and because of our reliance on NLP solvers that prefer smooth functions, we stay with our method of handling the objective. Also, eventually we hope to eliminate the need to introduce these binary variables, in which case our approach for the objective function would still be required. In any case, a detailed computational comparison between the fitted objective function and the discrete one is given at the end of Sect. 5.2.2.

Finally, we remark that in the preliminary report on our work (Bragalli et al. 2006), we described a different method for handling the discrete nature of the diameters. At that time, `Bonmin` was not yet able to handle SOS constraints, so we attempted to approximate the behavior of SOS branching via a different definition of binary variables and a judicious setting of branching priorities.

4.3 Parameterizing by area rather than diameter

We can use variables:

$$A(e) = \text{cross-sectional area of pipe } e \quad (\forall\, e \in E),$$

rather than the diameter variables $D(e)$ $(e \in E)$. This allows us to eliminate the nonlinearities and nonconvexities of the flow bounds which then become:

$$-v_{max}(e)A(e) \leq Q(e) \leq v_{max}(e)A(e) \quad (\forall e \in E).$$

The other constraints remain substantially similar. The simple bounds become:

$$\frac{\pi}{4} d_{min}^2(e) \leq A(e) \leq \frac{\pi}{4} d_{max}^2(e) \quad (\forall e \in E),$$

and the head loss across links constraints are:

$$H(i) - H(j) = \text{sgn}(Q(e))|Q(e)|^{1.852} \cdot 10.7 \cdot len(e) \cdot k(e)^{-1.852} \left(\frac{\pi}{4}\right)^{2.435} / A(e)^{2.435}$$

$(\forall\, e = (i, j) \in E).$

Although the head loss equations remain nonlinear, note that the effect of the use of areas instead of diameters is a perceptible reduction of the exponent of the design variables. Namely, we have $A(e)^{2.435}$ versus $D(e)^{4.87}$. Ultimately, however, the decision between the area and the diameter parameterization is, as usual, computational. We compared the two approaches with computational experiments and the diameter approach returned the same solution as the area one for 3 instances over 9, a better solution in other 3 instances, a worse one in one instance and numerical difficulties in the remaining 2 instances. Overall, the average of the percentage deviation[2] of the best MINLP solutions computed for the diameter approach with respect to those obtained with using the area is -0.16, i.e., the diameter parameterization is slightly better but less stable because we are not considering the 2 problematic instances. In addition, the average computing time to find the best solution for the 3 instances reporting the same solution (namely `shamir`, `hanoi` and `blacksburg`, described in Sect. 5.1) is 737 seconds for the area formulation, and 1,802 seconds for the diameter one. Although these results do not show a strict domination, we do prefer the area parameterization that appears ultimately more stable and better from a mathematical point of view. Thus, in Sect. 5.2.2 we report the results using the area parameterization.

5 Computational experience

In this section we give detailed computational results on instances from both the literature and real-world applications. In particular, we concentrate on the MINLP computation by discussing in detail the use of an open-source MINLP software, namely

[2] The percentage deviation of Algorithm A with respect to Algorithm B is computed as $100 \times (value[A] - value[B])/value[B]$.

Bonmin and the way in which we improved its performance on our nonconvex application.

Comparisons with results from the literature and a general evaluation of the quality of the MINLP solutions is reported in Sect. 6.

5.1 Instances

Our data comprises 9 instances that capture different aspects of real-world networks. In particular, these instances vary in size, type, number and diameter of the pipes that can be installed. Moreover, some special requirements to be discussed below are sometimes present.

The main characteristics of the instances are reported in Table 1.

For each instance, Table 1 reports the name and the numbers of junctions (including the reservoirs), reservoirs, pipes and diameter choices. Moreover, the column labeled "duplicates" indicates the number of pipes whose diameter is fixed but which can possibly be duplicated by installing a new pipe (whose diameter must be determined) in parallel. Finally, the last column indicates which currency is used to express the unit cost of the pipes, namely, US Dollar ($), Italian Lira (£) and Euro (€).

The instances shamir, hanoi, blacksburg and New York are taken from the literature, while the others are real-world instances of Italian water networks.[3]

For the instances from the literature, the only one that requires some preprocessing of the data in order to fit into our definitions is New York which will be separately discussed below. However, the data for the instance blacksburg available from Sherali et al. (2001) was incomplete, and the final version of the instance that we used and make available is certainly (slightly) different from the original one. In particular, the missing data are the diameter sizes and their costs. To complete the data of blacksburg, we then used diameters and costs from the instance shamir.

Table 1 Water networks

name	number of ...					unit
	junctions	reservoirs	pipes	duplicates	diameters	cost
shamir	7	1	8	–	14	$
hanoi	32	1	34	–	6	$
blacksburg	31	1	35	–	11	$
New York	20	1	21	21	12	$
foss_poly_0	37	1	58	–	7	£
foss_iron	37	1	58	–	13	€
foss_poly_1	37	1	58	–	22	€
pescara	71	3	99	–	13	€
modena	272	4	317	–	13	€

[3] All instances are available at www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm.

For the real-world instances, the three instances "foss_X" refer to a single neighborhood of Bologna, called Fossolo. The instance foss_poly_0 consists of the original data provided to us and the pipe material for that instance is polyethylene. The instance foss_iron is for the same network, but with almost twice as many choices of pipe diameters and with the material being cast iron. For the instance foss_poly_1 the material for the pipes is again polyethylene but there are more choices than foss_poly_0 for the pipe diameters.

The cost data for foss_poly_0 is out of date, and so the solution values cannot be directly compared to those of foss_poly_1 and foss_iron, which, in turn, can be reasonably compared. The value of the solution reported in Sect. 5.2 for foss_poly_1 is much lower than for foss_iron. At first this seems surprising, but it is because polyethylene is much cheaper than cast-iron.

Finally, pescara and modena are reduced versions of the water distribution networks of two medium-size Italian cities. The pipe material is cast iron and both costs and diameters are up-to-date values in the Italian market.

5.1.1 The famous New York instance

The New York instance was first introduced by Schaake and Lai (1969). The problem we need to solve for this instance is quite different from the original one. Given an existing network, the objective is to "renovate" it by considering the increase of the water demand due to the population growth. The existing network is no longer adequate for the increased demand, resulting in pressure violations at several junctions. Thus, the network must be modified by duplicating some of the pipes, i.e., putting new pipes in parallel with the existing ones, at a minimum cost.

The decisions one has to take are:

1. Select the pipes that need to be duplicated;
2. For each of these pipes, choose a diameter within the available diameter set.

In other words, with respect to our model, one has to add the null value to the diameter set: if such a null value is selected, it corresponds to the reverse of the decision 1 above, i.e., the pipe is not duplicated. However, such an explicit addition of the null diameter requires relevant modifications (consider the head loss equations), and an overall deterioration of our model. Thus, we decided to handle such a case by an alternative method, along a line proposed by Schaake and Lai (1969). Note that this approach was not presented and formally stated in Schaake and Lai (1969), but it can be read from the code reported in that paper. For the sake of clarity and completeness, we report it explicitly here.

The idea is to transform the problem, that includes the two decisions above, into our original problem, thus avoiding the first decision. We can easily do it introducing the *equivalent pipe* concept: We treat implicitly the two parallel pipes by means of a unique equivalent pipe that reproduces the same behavior at the extreme junctions of the pipe within the network. For each diameter of the duplicated pipe (including the null one) there is a discrete equivalent diameter associated with the pair existing/duplicated pipes.

We can prove the following simple result:

Theorem 2 *For each pipe $e \in E$ the new diameters and costs are, respectively*:

$$\mathfrak{D}_{new}(e,r) = \left(D_{fix}(e)^{\frac{4.87}{1.852}} + \mathfrak{D}(e,r)^{\frac{4.87}{1.852}}\right)^{\frac{1.852}{4.87}}$$
$$\mathfrak{C}_{new}(e,r) = \mathfrak{C}(e,r),$$

with $r = 0, 1, \ldots, r_e$ and where $D_{fix}(e)$ is the diameter of the existing pipe and $\mathfrak{D}(e,0) = \mathfrak{C}(e,0) = 0$.

Proof Formally, for each existing pipe $e \in E$, we add two pipes e' and e'' corresponding to the duplicated and equivalent pipes, respectively. First, note that the flow through the existing and duplicated pipes must follow the same direction because they have the same start and end junctions and, consequently, the same hydraulic head which determines the flow direction. Thus, $Q(e)$, $Q(e')$ and $Q(e'')$ agree in sign and denote the flows over the corresponding pipes. In order to impose the above described equivalence we must solve the following system of equations:

$$\begin{aligned}
&Q(e) + Q(e') = Q(e''),\\
&H(i) - 10.7 \cdot Q(e)^{1.852} \cdot k(e)^{-1.852} \cdot D(e)^{-4.87} \cdot len(e) - H(j) = 0,\\
&H(i) - 10.7 \cdot Q(e')^{1.852} \cdot k(e')^{-1.852} \cdot D(e')^{-4.87} \cdot len(e') - H(j) = 0,\\
&H(i) - 10.7 \cdot Q(e'')^{1.852} \cdot k(e'')^{-1.852} \cdot D(e'')^{-4.87} \cdot len(e'') - H(j) = 0.
\end{aligned}$$

As required, these equations guarantee that, substituting the two parallel pipes with the equivalent one, we obtain the same flow and the same head loss at the start and end junctions.

The system above can be easily simplified by substituting out the flows:

$$\left(\frac{H(i) - H(j)}{10.7 \cdot len(e)}\right)^{\frac{1}{1.852}} \cdot k(e) \cdot D(e)^{\frac{4.87}{1.852}} + \left(\frac{H(i) - H(j)}{10.7 \cdot len(e')}\right)^{\frac{1}{1.852}} \cdot k(e') \cdot D(e')^{\frac{4.87}{1.852}}$$
$$= \left(\frac{H(i) - H(j)}{10.7 \cdot len(e'')}\right)^{\frac{1}{1.852}} \cdot k(e'') \cdot D(e'')^{\frac{4.87}{1.852}}.$$

Because $len(e) = len(e') = len(e'')$ and, in this instance, $k(e) = k(e') = k(e'')$, it is easy to see that:

$$D(e'') = \left(D(e)^{\frac{4.87}{1.852}} + D(e')^{\frac{4.87}{1.852}}\right)^{\frac{1.852}{4.87}},$$

which proves the result. □

5.2 MINLP results

We have tested our approach by using (and modifying) the stable version the open-source MINLP solver `Bonmin` (see Bonami et al. 2008; Bonami and Lee 2006)

which is distributed on COIN-OR (Bonmin v. 1.0). In the following we describe the basic features of the solver and we report the computational results on the instances described in Sect. 5.1.

5.2.1 `Bonmin` *B&B algorithm description*

`Bonmin` (Basic Open-source Nonlinear Mixed INteger programming) is an open-source code for solving MINLP problems of the form:

$$\begin{aligned} \min \quad & f(x) \\ & g^L \leq g(x) \leq g^U \\ & x^L \leq x \leq x^U \\ & x \in \mathbb{R}^n \\ & x_i \in \mathbb{Z}, \quad \forall i \in I, \end{aligned}$$

where the functions $f : \{x \in \mathbb{R}^n : x^L \leq x \leq x^U\} \to \mathbb{R}$ and $g : \{x \in \mathbb{R}^n : x^L \leq x \leq x^U\} \to \mathbb{R}^m$ are assumed to be twice continuously differentiable, and $I \subseteq \{1, \ldots, n\}$.

There are several algorithms implemented within `Bonmin`:

B-BB, a simple branch-and-bound algorithm based on solving a continuous nonlinear program at each node of the search tree and branching on integer variables;
B-OA, an outer-approximation based decomposition algorithm;
B-QG, an outer-approximation based branch-and-bound algorithm;
B-Hyb, a hybrid outer-approximation/nonlinear programming based branch-and-cut algorithm.

The different methods that `Bonmin` implements are exact algorithms when the functions f and g are convex but are only heuristics when this is not the case. For an MINLP having a nonconvex relaxation (like the WDN problem), the B-BB algorithm should be used because the outer-approximation constraints are not necessarily valid inequalities for the problem. Although even B-BB is only a heuristic for such a nonconvex problem (the NLP problems at each node are not solved to global optimality), `Bonmin` includes several options tailored to improve the quality of the solutions it provides. First, in the context of nonconvex problems, `Ipopt` (the `Bonmin` default NLP solver) may end up in different local optima when started from different starting points. The two options `num_resolve_at_root` and `num_resolve_at_node` allow for solving the root node or each node of the tree, respectively, with a user-specified number of different randomly-chosen starting points, saving the best solution found. Note that the function to generate a random starting point is very naïve: it chooses a random point (uniformly) between the bounds provided for the variable.

Secondly, because the solution given by `Ipopt` does not truly give a lower bound, the user can adjust the fathoming rule to continue branching even if the solution value to the current node is worse than the best-known solution. This is achieved by setting `allowable_gap`, `allowable_fraction_gap` and `cutoff_decr` to negative values.

In the next section, we will describe how we used the options of Bonmin designed for nonconvex problems and some modifications we implemented in Bonmin specifically tailored for our WDN problem.

5.2.2 *Improving* Bonmin *for WDN problems*

As discussed in the previous section, Bonmin was originally developed for finding globally-optimal solutions to MINLPs having convex relaxations. However, some accommodations were made to handle nonconvex instances as well, already in the first released version, namely Bonmin (v. 0.1). In fact, these accommodations were developed and tested in the context of early stages of the present study. Additionally, we made and tested two further modifications, to better handle nonconvexities. We implemented these modifications by using the development version of Bonmin (the so called "trunk" version). As anticipated, the first of these two additional features (described in the following) has been already added to the current stable version of the software, namely Bonmin (v. 1.0). Eventually, the second modification may be adopted in a future release.

In particular, the following two main issues came up:

I.1 *Properly evaluating the objective value of integer feasible solutions*. In Sect. 3 we have introduced a new objective function so as to approximate the correct (discrete) one. During preliminary computational experiments, we noted that such an approximation sometimes has the effect of rejecting integer feasible solutions having the approximated objective value worse than the incumbent but with a better value with respect to the correct objective function. Such a behavior has been corrected by allowing Bonmin to work with *two objective functions*: the first one v_{fit} (i.e., the smooth continuous approximation) is used to guide the search, i.e., to fathom the nodes, while the second one v_{disc} (i.e., the correct *discrete* objective) is used to evaluate integer feasible solutions, so as to avoid disregarding improving leaves. So, each time a new integer feasible solution, say x^{new}, is found, the value of the discrete objective function is computed. In this way, we work with two incumbent solutions, say x^{inc} and $\bar{x}^{inc}$ for v_{fit} and v_{disc}, respectively. For every feasible solution, we do update separately each of the incumbents, if needed. More precisely, if $v_{fit}(x^{new}) \geq v_{fit}(x^{inc})$ using a single objective function would have led us to discard the solution x^{new}. In fact, we do test $v_{disc}(x^{new})$ with respect to $v_{disc}(\bar{x}^{inc})$ and in case $v_{disc}(x^{new}) < v_{disc}(\bar{x}^{inc})$ we do update the discrete incumbent. In any case, v_{fit} is used as primary objective function for fathoming fractional solutions at the nodes (see point I.2 below).

I.2 *Heuristically reducing the size of the search space*. The released versions of Bonmin statically define two parameters: The parameter cutoff is the value of a known feasible solution (i.e., the incumbent), while cutoff_decr is the value by which every new feasible solution should be improved with respect to the current incumbent (i.e., the cutoff above). On nonconvex problems, cutoff_decr is selected to be *negative* so as to act in a conservative manner with nodes whose continuous solution is not-too-much-worse than the current incum-

Table 2 Characteristics of the 50 continuous solutions at the root node

	mean	% dev. first	% dev. min	% dev. max	std. dev.	coeff. var.	# fail	# inf.
shamir	401,889.00	−4.880	−4.880	59.707	37,854.70	0.0941920	0	0
hanoi	6,134,520.00	−0.335	−1.989	2.516	91,833.70	0.0149700	0	0
blacksburg	114,163.00	1.205	−0.653	2.377	861.92	0.0075499	0	0
New York	82,646,700.00	0.605	−47.928	31.301	16,682,600.00	0.2018540	0	0
foss_poly_0	68,601,200.00	−1.607	−1.748	15.794	2,973,570.00	0.0433457	0	0
foss_iron	182,695.00	−2.686	−2.686	61.359	16,933.80	0.0926891	0	0
foss_poly_1	32,195.40	26.186	−17.193	42.108	4,592.63	0.1426490	0	0
pescara	1,937,180.00	−6.311	−6.596	54.368	274,956.00	0.1419370	0	0
modena	2,559,350.00	−0.254	−0.396	9.191	38,505.80	0.0150452	0	0

bent. However, we found out that such a static definition of `cutoff_decr` does not fit our purposes because it is hard to define a unique value for all instances. After preliminary computational testing, we implemented the following policy: the root node continuous value is computed for 50 different starting points and `cutoff_decr` is set as:

$$\texttt{cutoff\_decr} := -V \cdot f(\sigma), \tag{1}$$

where V is the average of the 50 root node continuous values, $\sigma \in [0, 1]$ is the coefficient of variation (standard deviation divided by the mean) of those values, and

$$f(\sigma) := \begin{cases} 0.02, & \text{if } \sigma < 0.1; \\ 0.05, & \text{if } \sigma \geq 0.1. \end{cases}$$

In other words, the parameter is set taking into account how much different the solutions at the root node are, so as to be more careful in fathoming a node when such a difference is large. The characteristics of the instances with respect to the 50 continuous solutions computed at the root node using different random starting points are given in Table 2. More precisely, Table 2 reports for each instance the mean (mean), percentage deviation of the first solution found (% dev first), percentage deviation of the minimum (% dev min) and the percentage deviation of the maximum (% dev max) value of the continuous solution at the root note over the 50 samples. The table then reports the standard deviation (std dev), the coefficient of variation (coeff var) and finally the number of failures of `Ipopt` (# fail) and the number of times the continuous problems turned out to be infeasible (# inf). Table 2 demonstrates that the way we have modeled the problem has a stable behavior, in the sense that the continuous solutions never have numerical difficulties nor turn out to be infeasible. On the other hand, the solution value depends a lot on the starting point, and the most unstable instances are `New York`, `foss_poly_1` and `pescara`.

Table 3 Computational results for the MINLP model (part 1). Time limit 7200 seconds

	$v_{fit}(x^{best})$	$v_{disc}(\bar{x}^{best})$	% dev. $v_{fit}(\bar{x}^{best})$	
`shamir`	400,749.77	419,000.00	0.000	
`hanoi`	6,109,840.00	6,109,620.90	0.000	
`blacksburg`	118,251.06	118,251.09	0.000	
`New York`	39,569,920.48	39,307,799.72	0.550	$\surd$
`foss_poly_0`	69,161,700.00	70,680,507.90	0.000	
`foss_iron`	179,134.00	178,494.14	0.015	$\surd$
`foss_poly_1`	29,016.60	29,117.04	0.000	
`pescara`	1,850,720.00	1,820,263.72	0.193	$\surd$
`modena`	2,609,550.00	2,576,589.00	0.341	$\surd$

5.2.3 MINLP solutions

The results obtained using Bonmin (v. 1.0) are reported in Tables 3 and 4, running the code with a time limit of 7,200 CPU seconds on a single processor of an Intel Core2 CPU 6600, 2.40 GHz, 1.94 GB of RAM under Linux. In particular, Table 3 reports the best solution values computed for the instances in the testbed. The value of the best solution with the fitted objective function (denoted as $v_{fit}(x^{best})$) is compared with the value of the best solution found with respect to the true objective function (denoted as $v_{disc}(\bar{x}^{best})$). In addition, we report the percentage deviation of the value of the solution $\bar{x}^{best}$ once mapped on the fitted objective function (denoted as % dev $v_{fit}(\bar{x}^{best})$). In particular, we marked (with a "$\surd$") in the last column the four instances for which values $v_{fit}(x^{best})$ and $v_{fit}(\bar{x}^{best})$ are different, i.e., the instances in which the use of both objective functions simultaneously had a very positive effect. Note that a positive value for the percentage deviation indicates an improvement: the (true) best solution would not have been found without modification I.1 because its fitted value was in fact worse than the fitted value of the incumbent used by the algorithm. Table 4 reports additional results on the same instances and with the same tuning of the code. In particular, besides the best MINLP solution value with respect to the true objective function ($v_{disc}(\bar{x}^{best})$) (same column of Table 3), we report the CPU time in seconds to find such a solution (time) and percentage deviation of the best MINLP solution value after 1,200 CPU seconds (denoted as % dev $v_{disc}(\bar{x}^{first})$, recall that the overall time limit is 7,200 CPU seconds). Finally, the last two columns report the number of updates of the incumbent solution value for the fitted (# fit) and true (# true) objective function, respectively. When such numbers are different for the same instance, it means that using simultaneously two objective functions had an effect, i.e., it changed the explored tree. Such instances are a superset of the four marked in Table 3 with a "$\surd$" in the last column. Precisely, the effect of modification I.1 above is crucial for the four instances `New York`, `foss_iron`, `pescara` and `modena`, in which the final solution of the fitted objective function is not the best one with respect to the discrete objective function. Moreover, Table 4 demonstrates that besides the four above instances, the use of the two objective functions also has an

Table 4 Computational results for the MINLP model (part 2). Time limit 7200 seconds

	$v_{disc}(\bar{x}^{best})$	time	% dev. $v_{disc}(\bar{x}^{first})$	# fit	# true
shamir	419,000.00	1	0.000	2	2
hanoi	6,109,620.90	191	0.000	6	5
blacksburg	118,251.09	2,018	0.178	6	6
New York	39,307,799.72	5	0.000	5	6
foss_poly_0	70,680,507.90	41	0.000	3	3
foss_iron	178,494.14	464	0.000	11	11
foss_poly_1	29,117.04	2,589	0.119	5	5
pescara	1,820,263.72	2,084	0.724	7	8
modena	2,576,589.00	3,935	0.055	4	3

impact on the hanoi instance where, during the search, some solutions with good value of the discrete objective function are kept. These solutions are not the best ones at the end of the 2 hours time limit, but clearly they could have been with a different time limit.

The effect of modification I.2 is, instead, an improvement for the instance foss_poly_1: specifically, the solution significantly improves from 29,151.70 to 29,117.04.

Note that the solutions obtained within 20 minutes of CPU time (see Table 4) are also very good and show how the MINLP search is effective also for short computing times.

The overall behavior of the code is dependent on the search options that can be selected among the Bonmin arsenal. In particular, the reported results are all obtained with tree_search_strategy = dive and node_comparison = best-bound which turned out to give the most stable and effective version. On the other hand, slightly better results on single instances can be obtained with different parameters and in particular using node_comparison = dynamic.

A natural computational question is the impact of using the fitted objective function as compared to using only the discrete one. We performed an additional set of experiments with the original discrete function and the results are reported in Table 5. For the discrete case, we report in Table 5 the percentage deviation of the best solution (denoted as % dev "discrete") with respect to our best results $v_{disc}(\bar{x}^{best})$ and the corresponding computing time to achieve the best solution. The table shows that on five of the nine instances, the final solution using the discrete diameters is the same, but the average computing time to find those solutions is 2,135.5 seconds as compared to 535.8, i.e., it requires more time on average. For the four remaining instances, the fitted objective approach provides a better solution than the one obtained using the original discrete function. In summary, the algorithm using the discrete objective function performs somewhat worse than the one with the fitted one, and in particular this happens for the largest instances. This behavior is no surprise: in the discrete case the continuous solution obtained by relaxing integrality is always much smaller than the one for the fitted objective function because the piecewise-linear function can use

Table 5 Computational results for the MINLP model comparing the fitted and the discrete objective functions. Time limit 7200 seconds

	$v_{disc}(\bar{x}^{best})$	time	% dev. "discrete"	time
shamir	419,000.00	1	0.00	4
hanoi	6,109,620.90	191	0.00	1,378
blacksburg	118,251.09	2,018	0.00	1,936
New York	39,307,799.72	5	0.00	198
foss_poly_0	70,680,507.90	41	0.57	5,514
foss_iron	178,494.14	464	0.00	7,161
foss_poly_1	29,117.04	2,589	0.18	2,811
pescara	1,820,263.72	2,084	2.12	690
modena	2,576,589.00	3,935	1.94	167

non-consecutive discrete values in the convex combination. This results in a much larger search space to be explored because bad nodes might not be fathomed.

Finally, concerning the time limit of 7,200 CPU seconds, we note that it is reached in all reported tests with different settings and options, the only exception being instance shamir.

6 Evaluating the quality of the MINLP solutions

As explained in the previous sections, our aim has always been to be able to model in a mathematically accurate way the real-world application at hand and simultaneously finding "good" solutions in short computing times. Of course, there are several ways of defining what "good" means and in the following we will consider four of them. More precisely, in Sect. 6.1 we will discuss the computation of valid lower bounds by using the Global Optimization software Baron (see, Tawarmalani and Sahinidis 2004). In Sect. 6.2 we will compare our MINLP solutions with others in the literature obtained mainly by meta-heuristic methods. In Sect. 6.3 we will discuss a standard linearization method to handle the nonlinear part of the problem, i.e., the Hazen-Williams equations. Finally, in Sect. 6.4 we will show some interesting properties of our solutions that make them immediately usable by practitioners.

6.1 Computing valid lower bounds with Baron

Currently Baron (see, Tawarmalani and Sahinidis 2004; Sahinidis and Tawarmalani 2005) is best software for global optimization. It uses: (i) under and over estimators on low-dimensional functions to compute valid lower bounds for nonconvex NLPs obtained by dropping the integrality requirements of nonconvex MINLPs, and (ii) spatial branch-and-bound to iteratively improve the estimation quality and enforce integrality.

Table 6 Computational results for the MINLP model comparing Baron and Bonmin (from Table 3)

	Baron				Bonmin	
	time limit 2 CPU hours		time limit 12 CPU hours			
	UB	LB	UB	LB	$v_{disc}(\tilde{x}^{best})$	% gap
shamir	419,000.00	419,000.00	419,000.00	419,000.00	419,000.00	0.00
hanoi	6,309,727.80	5,643,490.00	6,219,567.80	5,783,950.00	6,109,620.90	5.63
blacksburg	n.a.	55,791.90	n.a.	105,464.00	118,251.09	12.12
New York	43,821,000.00	29,174,000.00	43,821,000.00	29,174,000.00	39,307,799.72	34.74
foss_poly_0	n.a.	64,787,300.00	n.a.	64,787,300.00	70,680,507.90	9.10
foss_iron	n.a.	170,580.00	n.a.	170,580.00	178,494.14	4.64
foss_poly_1	n.a.	25,308.20	n.a.	25,308.20	29,117.04	15.05
pescara	n.a.	1,512,640.00	n.a.	1,512,640.00	1,820,263.72	20.34
modena	n.a.	2,073,050.00	n.a.	2,073,050.00	2,576,589.00	24.29

Although global optimization is not in the scope of our work, we tested both our instances from the literature and real-world instances with Baron v.8.5.1 to evaluate the difficulty of these instances and the quality of our solutions.

The results are reported in Table 6 where we used two different time limits of 2 and 12 CPU hours on an Intel Pentium IV with 3.00 GHz, 1 GB RAM under Linux. The reason for using two different time limits was to understand the progresses of the code, especially on the real-world instances (the lower part of the table). For each of the instances Table 6 reports for Baron the best upper (UB) and lower (LB) bounds for each time limit, and for Bonmin the best solution (as reported in Table 3), i.e., an upper bound, within 2 CPU hours. The last column indicates the % gap of the feasible solution obtained by Bonmin relative to the best lower bound computed by Baron. Entries marked with "n.a." indicate "not available".

The results show that, except for three of the (smaller) literature instances, Baron is unable to find any feasible solution, and for all (larger) real-world instances, the lower bound after 12 hours has not improved compared to the one obtained after only 2 hours. Baron is able to solve the instance shamir to optimality, while the upper bounds for the instances hanoi and New York are worse than the ones computed by Bonmin.

The above results are not surprising: (i) we applied Baron as a black-box without parameter tuning, and (ii) to compute valid bounds and prove optimality Baron has to explore a much larger solution space—too large at the moment—for real-world instances. Nevertheless, the % gaps of Bonmin upper bounds are very reasonable with a maximum for the instance New York that has, as we already discussed, a very special structure.

To give the reader an idea of how good the solutions we propose are, we report in Table 7 the global optimum of the split-pipe relaxation for instances hanoi and New York as computed by Sherali et al. (2001). We excluded shamir because we know the global optimum. We excluded blacksburg because the data on the diameters are different, thus the solution of the split-pipe approach is not comparable with our solution. The lower bound given by the optimal solution of the split-pipe relaxation

Table 7 Global optimum of the split-pipe relaxation

	Baron	Bonmin	
	LB	$v_{disc}(\bar{x}^{best})$	% gap
hanoi	6,055,536.31	6,109,620.90	0.89
New York	37,878,580.89	39,307,799.72	3.77

is stronger than the one provided by Baron, and it is clear that for both instances the solutions we compute within the time limit are of very good quality.

6.2 Literature comparison

The comparison with "existing numbers", i.e., with previous known solutions for the benchmark instances, is in general very difficult because of different parameter sets and coefficients used for the Hazen-Williams formula (see, e.g., the discussion in Savic and Walters 1997). Despite such a difficulty, we report the following results.

1. For the small instance shamir, we find the optimal (as shown by the results of Baron discussed in the previous section) solution.
2. On instance blacksburg we are not able to compare our results because: (i) the only results are those reported in Sherali et al. (2001) which are obtained with the split-pipe approach and (ii) part of the data was missing and, as mentioned in Sect. 5.1, the instance we use is different from the one used in that paper.
3. As stated in Sect. 3, the set of coefficients we decided to use for the Hazen-Williams equation is the one of Walski (1984). In order to provide an informative comparison with other authors on the remaining instances, namely hanoi and New York, we ran our code by using each time the coefficient set proposed in the paper to be compared. In particular, in Table 8 we compare our MINLP approach with the approaches in the following papers:

   - Savic and Walters (1997). In such a paper the authors analyze the results previously reported in the literature for the problem and define two different sets of coefficients corresponding essentially to the most "relaxed" version of the empirical formula for the Hazen-Williams equation and to the most "restrictive" one. We consider in Table 8 both versions, denoted as "SW99 rel." and "SW99 res.", respectively.
   - Dandy et al. (1996), denoted as "DSM96".
   - Cunha and Sousa (1999), denoted as "CS99".

   The MINLP solution obtained by our model using the above sets of coefficients is as usual denoted as $v_{disc}(\bar{x}^{best})$ in Table 8. An entry "–" in the table denotes that a particular instance has not been considered in a specific paper.

   Table 8 shows that our MINLP approach is able to find a solution at least as good as the other approaches in all but one of the cases. More precisely, it improves three times, it matches the same result twice and it is slightly worse compared to "SW99 res." on instance New York. This is a very satisfactory behavior which proves that the approach is not affected by the coefficient sets used for the Hazen-Williams equation.

Table 8 MINLP results compared with literature results

	Savic and Walters (1997)		Savic and Walters (1997)		Cunha and Sousa (1999)	
	SW99 rel.	$v_{disc}(\bar{x}^{best})$	SW99 res.	$v_{disc}(\bar{x}^{best})$	CS99	$v_{disc}(\bar{x}^{best})$
`hanoi`	6.073 e+06	6.066 e+06	6.195 e+06	6.183 e+06	6.056 e+06	6.056 e+06
	Savic and Walters (1997)		Savic and Walters (1997)		Dandy et al. (1996)	
	SW99 rel.	$v_{disc}(\bar{x}^{best})$	SW99 res.	$v_{disc}(\bar{x}^{best})$	DSM96	$v_{disc}(\bar{x}^{best})$
`New York`	37.13 e+06	36.38 e+06	40.42 e+06	40.47 e+06	38.8 e+06	38.8 e+06

Concerning the running times, in Savic and Walters (1997), the reported results are obtained by several runs of 3 hours for each instance on a PC 486/DX2 50 computer. In Dandy et al. (1996), each run is of 50 minutes on a Sun Sparc 1 + Station with the operating system SunOS 4.1. Finally, in Cunha and Sousa (1999), the runs are of 2 hours on a Pentium PC at 166 MHz.

It is hard to compare these results because of the different computing platforms but our algorithm is able to find the solutions for all sets of coefficients of `hanoi` and `New York` within 1 minute of CPU time, thus showing a very competitive behavior.

6.3 MILP results

The only details that are needed to implement the Artina and Walker (1983) MILP approach for our problem concern the piecewise-linear approximation of the Hazen-Williams equations. First, note that for every $e = (i, j) \in E$, we have to separately consider the case in which the flow goes from i to j or vice versa. In other words, for approximating the two parts of the curve described by the Hazen-Williams equation, we need two separate sets of weights which are then combined by writing a unique SOS constraint of type 2.

Second, for each of the two parts above, say from i to j, we have to decide how to sample the curve so as to approximate it. In particular, for a fixed diameter value, we plot the flow on the y-axis as a function of the head loss $H(i) - H(j)$, on the x-axis. Then, we compute an upper bound on the head loss value as:

$$\Delta H_{ij}(e) = \min\Bigg\{\max\big\{(ph_{max}(i) + elev(i)) - (ph_{min}(j) + elev(j)), 0\big\}, \max_{r=1,\ldots,r_e}\left\{\frac{10.7 \cdot len(e)}{k(e)^{1.852} \cdot \mathfrak{D}(e,r)^{4.87}}\left(\frac{\pi}{4}\mathfrak{D}(e,r)^2 v_{max}(e)\right)^{1.852}\right\}\Bigg\}.$$

The second term of the above equation can be simplified by recalling that $\mathfrak{D}(e, 1) < \mathfrak{D}(e, 2) < \cdots < \mathfrak{D}(e, r_e)$, and the bound can be rewritten as:

$$\Delta H_{ij}(e) = \min\Bigg\{\max\big\{(H_{max}(i) - H_{min}(j)), 0\big\}, \frac{10.7 \cdot len(e)}{k(e)^{1.852} \cdot d_{min}(e)^{4.87}}\left(\frac{\pi}{4}d_{min}(e)^2 v_{max}(e)\right)^{1.852}\Bigg\},$$

where $H_{max}(i) := ph_{max}(i) + elev(i)$ and $H_{min}(j) := ph_{min}(j) + elev(j)$.

The obtained interval $[0, \Delta H_{ij}(e)]$ is then split in two parts $[0, \frac{1}{3}\Delta H_{ij}(e)]$ and $[\frac{1}{3}\Delta H_{ij}(e), \Delta H_{ij}(e)]$. Within such intervals we perform uniform sampling by using the same number of linearization points. This means, of course, that we have a better approximation in the first part of the interval which is sensible because in the second part the curve is more flat, thus easy to approximate well with few points.

Note that, the analogous upper bound computed in the reverse direction from j to i, $\Delta H_{ji}(e)$, only differs in the first term above. Moreover, both bounds $\Delta H_{ij}(e)$ and $\Delta H_{ji}(e)$ are constant with respect to the diameter, i.e., the maximum value of the head loss on the x-axis is the same for every curve obtained by fixing the diameter value. In other words, the x-axis values of the linearization points are common to each diameter, while the corresponding y-axis value changes.

The computational results, obtained by using such an MILP model and Ilog-Cplex (v. 10.1) as MILP solver, are disappointing—in fact, this was our motivation for trying an MINLP approach. The MILP problems are difficult to solve mainly because once the diameters have been settled, the objective function is constant, and the model reduces to a feasibility problem which approximates a pure NLP. It is not surprising, then, that finding a feasible solution of a system of nonlinear equalities with a standard MILP technique is not a good idea. Note that, when the diameters/areas are fixed, the feasibility problem reduces to a system of $|E| + |N \backslash S|$ equations in $|E| + |N \backslash S|$ variables, plus, in our model, there are inequalities corresponding to the bounds on the variables which have to be satisfied.

Moreover, the MILP approach is heavily dependent on the quality of the approximation of the nonlinear component. If such an approximation is of high quality (i.e., the number of linearization points is large), no MILP feasible solution (i.e., a solution which satisfies with the default tolerance of 10^{-6} the constraints of the MILP model) can be found by Ilog-Cplex (v. 10.1) within reasonable CPU times: from a few hours for the small networks up to days for the larger ones. In other words, the models are not useful because they cannot find even feasible solutions in reasonable computing time.

Otherwise, with a rough approximation, the situation becomes even more complicated. Let us consider for example instance `hanoi`. The MILP model obtained with 14 linearization points (7 for each direction i to j and j to i) is solved to optimality by Ilog-Cplex (v. 10.1) in around 40 CPU minutes, and the solution has value 6,170,269. Such a solution is worse than the one obtained with the MINLP approach (6,109,620.90), and it is slightly NLP infeasible once the corresponding set of diameters is given to the NLP solver `Ipopt`. In addition, we fixed the diameters corresponding to the MINLP solution into the MILP model and realized that the solution is indeed infeasible for such a rough approximation. In fact, this set of diameters becomes feasible only using at least 170 (!!) linearization points. However, as mentioned above, solving the complete MILP model obtained using 170 points is out of the question even for a small-/medium-size instance as `hanoi`.

The trend outlined above is confirmed on the other 8 instances in our data set. Going from the smallest to the biggest, the only instance for which the MILP approach is effective is `shamir` which is small and easy enough to be solved with good accuracy and quality by using 14 linearization points; the value of 419,000 is proven to be optimal in a few seconds. For `blacksburg`,

which is still pretty small, we used 30 linearization points, and the first feasible solution (with the discussed approximation) is obtained after 2 hours of CPU time. The best solution found within a time limit of 48 CPU hours has value 129,280.60, which is larger than the best MINLP solution of value 118,251.09. Approximating this instance seems to be rather hard; the MINLP diameter set is not feasible for the MILP model even allowing 4,000 linearization points. For `New York`, we used 30 linearization points, and the first feasible solution is obtained after 3 minutes, but its value is quite bad (61,382,605.6). After 2 CPU hours, the best solution value is 43,317,003.3 which becomes 40,474,098.3 after 2 days. Anyway the best solution found by the MINLP approach is not feasible for the MILP approximation considering less than 90 linearization points. If we run the MILP model with 90 linearization points, we are able to find the first feasible solution after more than 20 minutes (1,479 seconds, value 65,819,089.9), but after 3 hours we have still a quite bad solution (value 46,045,781.3). Even worse results are obtained for the real-world instances in the `fossolo` set and for `pescara` and `modena`.

It is reasonable to ask if these disappointing MILP results depend on the tolerance/accuracy required by the MILP solver, which might be too high compared to those of the MINLP solvers. We extensively experimented with different tolerance values by also using as a guidance the "FeasOpt" feature of Ilog-Cplex (v. 10.1). Indeed, "FeasOpt" heuristically gives a measure of the amount of relaxation of the constraints (in terms of sum of the artificial variables introduced to make the constraints slack) that is needed to make a model feasible. Essentially, one either increases the quality of the approximation with more linearization points or decreases the integer and feasibility tolerance of the solver. However, the results in the latter case might not be very meaningful. For example, for the `New York` instance, to make the best MINLP solution feasible for MILP using only 14 linearization points, one needs to set the integer tolerance to 10^{-1}; with, instead, 50 linearization points, an integer tolerance of 10^{-2} suffices. Such very large tolerances admit not only our (truly feasible) MINLP solutions as feasible solution, but they also allow many truly infeasible configurations which have to be discarded *a posteriori*, thus making the overall approach not practical.

6.4 Practical use of the MINLP solutions

The analysis of the best MINLP solutions for the considered instances shows configurations in which the size of the selected diameters decreases from the reservoir toward the parts of the network farther away from the inlet point. This characteristic of the allocation of diameters to pipes plays in favor of a correct hydraulic operation of the network and has a beneficial effect on water quality, see, e.g., the discussion in Van Den Boomen et al. (2004). This is depicted in Fig. 2 where the size of each diameter is proportional to the thickness of the arc[4]. It is easy to see that there are no

[4] In Fig. 2 diameters are expressed in meters, and the diameter is equal to 0.06 for the pipes without explicit number, i.e., the minimum diameter permissible for this data set.

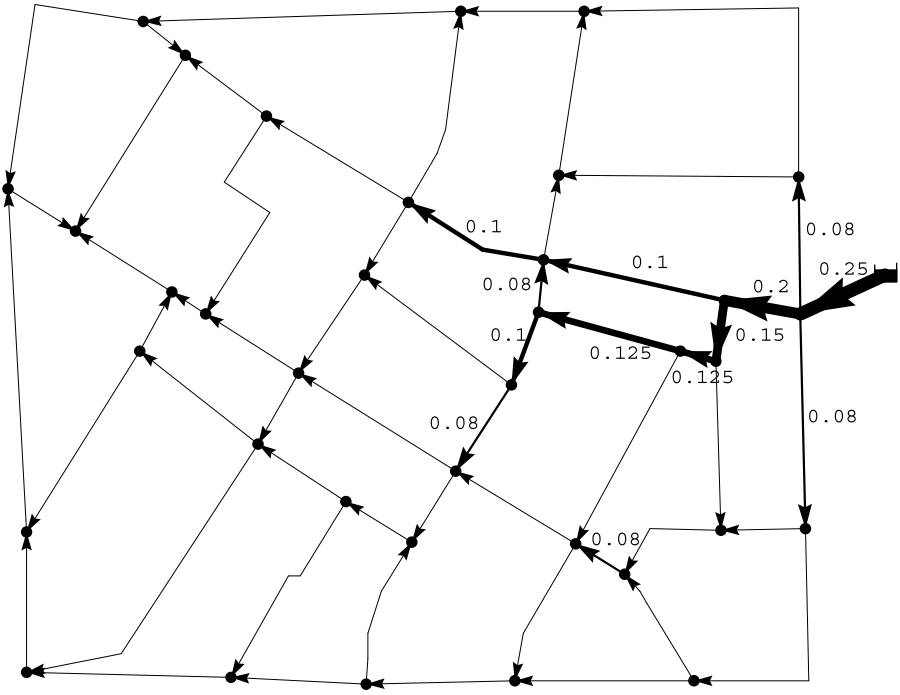

Fig. 2 Solution for Fossolo network, version `foss_iron`

pipes having large diameters isolated in the network. Such a characteristic, very appreciated by practitioners, does not normally occur when the design is done by a sort of "generate and test" approach in which configurations are produced within a naïve genetic-algorithms framework and then simulated for feasibility by using EPANET (EPANET v. 2.0).[5] However, that is the approach commonly used in practice, and it requires considerable postprocessing for correcting configurations that are trivially non-optimal with pipes having diameter significantly different from the surrounding ones. In such a postprocessing, the arcs might be analyzed one-by-one in the attempt of reducing the size of their assigned diameters. We denote as *1-optimal* a solution in which the size of the diameter $\mathfrak{D}(e, i)$ of a single pipe e cannot be reduced to the value $\mathfrak{D}(e, i-1)$. Instead, the structure of the solutions obtained by our algorithm reflects such a general design criterion widely used by practitioners and represents an interesting indicator of the strength of the proposed approach. The solution depicted in Fig. 2 is ready to be used in practice without any postprocessing, i.e., it is 1-optimal. In practice, it never happened in our large set of experiments that the algorithm returned as a best incumbent a non 1-optimal solution.[6]

[5] EPANET is free software distributed by the US Environmental Protection Agency.

[6] The solutions of the entire testbed can be found in the same repository of the instances.

7 Conclusions

In this paper we have been able to get effective solutions, both in terms of quality and accuracy, to practical instances of water-network optimization problems. Although Mixed Integer Linear Programming models were known since the 80s for the problem, those models are very difficult to solve by sophisticated MILP solvers because they are somehow unnatural. A much more natural Mixed Integer Non Linear Programming formulation allowed us to find the above mentioned good solutions in very reasonable computing times. In addition, these solutions are immediately usable in practice (as discussed in Sect. 6.4) because they are characterized by an allocation of diameters to pipes that leads to a correct hydraulic operation of the network. This is not the case for most of the other methods presented in the literature.

These successful results were achieved in two stages:

1. In a first phase, we could obtain from reasonable to good results with very low development time mainly because of the availability of software for finding good solutions to MINLP problems and the easy interface to such software via the modeling language AMPL.
2. In a second phase, we moved to a more sophisticated analysis of both the model and the algorithm, and we have been able to improve over the initial results significantly by using special-purpose modeling tricks and by contributing to the open-source platform provided by the software `Bonmin` with effective adaptations to deal with nonconvex MINLPs and multiple objective functions. These algorithmic ideas developed in the context of this work are now mostly part of the current version of `Bonmin` and this is a significant methodological contribution of our paper.

The code developed in this project is part of the current release of Bonmin (v. 1.0) for further use in different applications. Our belief is that such a success can be obtained within the same framework for other instances of optimization problems having significant discrete and nonlinear aspects.

Acknowledgements We thank Pierre Bonami and Andreas Wächter for helping us to use `Bonmin` in the best possible way. The last two authors are partially supported by "Ministero dell'Università e della Ricerca" (MIUR), Italy. We thank Stefan Vigerske for interesting discussions on the subject and for helping us to compute the lower bounds with `Baron`. We finally thank two anonymous referees for useful comments that helped improve the paper presentation.

References

Artina S, Walker J (1983) Sull'uso della programmazione a valori misti nel dimensionamento di costo minimo di reti in pressione. Atti Accad Sci Ist Bologna, Serie III, vol 271(X) (in Italian)

Beale E, Tomlin J (1970) Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In: Lawrence J (ed) Proc of the 5th int conf on operations research, pp 447–454

Bonami P, Lee J (2006) Bonmin users' manual. Tech rep

Bonami P, Biegler L, Conn A, Cornuéjols G, Grossmann C, Laird I, Lee J, Lodi A, Margot F, Sawaya N, Wächter A (2008) An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optim 5:186–204

Bonmin (v. 0.1) https://projects.coin-or.org/Bonmin
Bonmin (v. 1.0) https://projects.coin-or.org/Bonmin
Bragalli C, D'Ambrosio C, Lee J, Lodi A, Toth P (2006) An MINLP solution method for a water network problem. In: Azar Y, Erlebach T (eds) Algorithms—ESA 2006. Lecture Notes in Computer Science, vol 4168. Springer, Berlin, pp 696–707
Cunha M, Sousa J (1999) Water distribution network design optimization: simulated annealing approach. J Water Resour Plan Manag, Div Soc Civ Eng 125:215–221
Dandy G, Simpson A, Murphy L (1996) An improved genetic algorithm for pipe network optimization. Water Resour Res 32:449–458
Eiger G, Shamir U, Ben-Tal A (1994) Optimal design of water distribution networks. Water Resour Res 30:2637–2646
EPANET (v. 2.0) www.epa.gov/ORD/NRMRL/wswrd/epanet.html
Fourer R, Gay D, Kernighan B (2003) AMPL: a modeling language for mathematical programming, 2nd edn. Duxbury Press/Brooks/Cole Publishing Co., N. Scituate
Fujiwara O, Khang D (1990) A two-phase decomposition method for optimal design of looped water distribution networks. Water Resour Res 26:539–549
Ilog-Cplex (v. 10.1) www.ilog.com/products/cplex
Ipopt (v. 3.5) https://projects.coin-or.org/Ipopt
Lansey K, Mays L (1989) Optimization model for water distribution system design. J Hydraul Eng 115:1401–1418
Leyffer S (April 1998; revised March 1999) User manual for `MINLP_BB`. Tech rep, University of Dundee
Mathematica (v. 7.0) www.wolfram.com/products/mathematica/index.html
NEOS (v. 5.0) www-neos.mcs.anl.gov/neos
Sahinidis NV, Tawarmalani M (2005) BARON 7.2.5: global optimization of mixed-integer nonlinear programs, User's Manual
Savic DA, Walters GA (1997) Genetic algorithms for the least-cost design of water distribution networks. J Water Resour Plan Manag 123:67–77
Schaake JCJ, Lai D (1969) Liner programming and dynamic programming application to water distribution network design. Report 116, Hydrodynamics Laboratory, Department of Civil Engineering, School of Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts
Sherali HD, Subramanian S, Loganathan GV (2001) Effective relaxation and partitioning schemes for solving water distribution network design problems to global optimality. J Glob Optim 19:1–26
Tawarmalani M, Sahinidis NV (2004) Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math Program 99:563–591
Van Den Boomen M, Mazijk AV, Beuken R (2004) First evaluation of new design concepts for self-cleaning distribution networks. J Water Supply, Res Technol, AQUA 53(1):43–50
Walski T (1984) Analysis of water distribution systems. Van Nostrand Reinhold Company, New York
Walski T, Chase D, Savic D (2001) Water distribution modeling. Haestad Methods, Waterbury
Xu C, Goulter I (1999) Reliability-based optimal design of water distribution networks. J Water Resour Plan Manag 125:352–362