RC24495 (W0802-056) February 13, 2008
Mathematics

IBM Research Report

Water Network Design by MINLP

Cristiana Bragalli[1], Claudia D'Ambrosio[2], Jon Lee[3], Andrea Lodi[2], Paolo Toth[2]

[1]DISTART
University of Bologna
viale Risorgimento 2
40136 Bologna
Italy

[2]DEIS
University of Bologna
viale Risorgimento 2
40136 Bologna
Italy

[3]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

Water Network Design by MINLP

Cristiana Bragalli
DISTART, University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy, cristiana.bragalli@mail.ing.unibo.it

Claudia D'Ambrosio
DEIS, University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy, c.dambrosio@unibo.it

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

Andrea Lodi, Paolo Toth
DEIS, University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy, {andrea.lodi@unibo.it, paolo.toth@unibo.it}

We propose a solution method for a water-network optimization problem 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.

Key words: mixed integer nonlinear programming; modeling; open-source software; real-world instances.
History:

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.

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 §3. 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 encountered, 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 Sven Leyffer's code `MINLP_BB` (Leyffer (April 1998; revised March 1999), available from the University of Dundee) as well as the — at that time new — CMU/IBM open-source MINLP code `Bonmin v. 0.1` (Bonami et al. (2005), Bonami and Lee (June 2006), available from COIN-OR). In fact, it was in the context of these investigations that Bonmin was adapted for use on nonconvex MINLP problems.

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

In §1, we formally set the notation for specifying instances of the problem. In §2, we describe the problem more fully, through a preliminary continuous model. In §3.1, we survey earlier approaches, while in §3.2 we describe an NLP model in which we make a smooth (approximate) relaxation of the preliminary model described in §2, so that we can apply methods of smooth optimization. In §3.3, we describe how we incorporate binary variables for the purposes of then applying MINLP codes. In §3.4, so as to decrease the nonlinearity and nonconvexity, we describe a reparameterization of pipes by (cross-sectional) area, rather than diameter. In §4, we describe the results of computational experiments. Finally, in §5 we draw some conclusions.

1 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 ($S \subset N$).

$\delta_+(i)$ = 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 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:

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

2 A preliminary continuous model

In this section, we fully 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 ($e \in E$).

$D(e)$ = diameter of pipe e ($e \in E$).

$H(i)$ = hydraulic head of junction i ($i \in N$).

Simple bounds [Linear]:

$d_{min}(e) \leq D(e) \leq d_{max}(e) \quad (\forall e \in E)$.

$ph_{min}(i) + elev(i) \leq H(i) \leq ph_{max}(i) + elev(i) \quad (\forall\ i \in N \setminus S)$.

$H(i) = h_s(i) \quad (\forall\ i \in S)$.

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 gauge pressure head (see Walski et al. (2001)).

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

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

Flow conservation [Linear]:

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

Head loss across links [Nonsmooth and nonconvex]:

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

This last constraint models pressure loss in water pipes due to friction using the empirical Hazen-Williams equation. This is an accepted model for fully turbulent flow in *water* networks (see Walski (1984)). Diameter is bounded away from 0, so the only nondifferentiability is when the flow is 0.

Objective to be minimized [Discrete]:

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

Since 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 — which is appreciated by an NLP solver — to closely fit the discrete cost data.

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 arc e minimizes:

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

where t is the desired degree of the polynomial.

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.

To find the value of the coefficients for each piece of the cubic spline, we solve the following system of equations (for each pipe e):

$$\begin{aligned}
(a_r(e) + b_r(e)\ \mathfrak{D}(e,r) + c_r(e)\ \mathfrak{D}(e,r)^2 + d_r(e)\ \mathfrak{D}(e,r)^3) - \mathfrak{C}(e,r-1) &= 0\ , \quad r = 2,\ldots,r_e\ ; \\
(a_r(e) + b_r(e)\ \mathfrak{D}(e,r) + c_r(e)\ \mathfrak{D}(e,r)^2 + d_r(e)\ \mathfrak{D}(e,r)^3) - \mathfrak{C}(e,r) &= 0\ , \quad r = 2,\ldots,r_e\ ; \\
(b_r(e) + 2\ c_r(e)\ \mathfrak{D}(e,r) + 3\ d_r v\ \mathfrak{D}(e,r)^2) - & \\
(b_{r+1}(e) + 2\ c_{r+1}(e)\ \mathfrak{D}(e,r) + 3\ d_{r+1}(e)\ \mathfrak{D}(e,r)^2) &= 0\ , \quad r = 2,\ldots,r_e-1\ ; \\
(2\ c_r(e) + 6\ d_r(e)\ \mathfrak{D}(e,r)) - (2\ c_{r+1}(e) + 6\ d_{r+1}(e)\ \mathfrak{D}(e,r)) &= 0\ , \quad r = 2,\ldots,r_e-1\ ; \\
2\ c_2(e) + 6\ d_2(e)\ \mathfrak{D}(e,1) &= 0\ ; \\
2\ c_{r_e}(e) + 6\ d_{r_e}(e)\ \mathfrak{D}(e,r_e) &= 0\ ,
\end{aligned}$$

where the first two equations insure that the cubic spline fits the discrete costs, the third and the forth equations insure that the value of two consecutive pieces of the cubic spline have the same first and second derivative at the point of juncture, and the last two equations insure that the second derivative of the cubic spline is equal to zero at the points $(\mathfrak{D}(e,1),\mathfrak{C}(e,1))$ and $(\mathfrak{D}(e,r_e),\ \mathfrak{C}(e,r_e))$. Then the continuous objective function is defined in this way:

$$\begin{aligned}
\min \quad & \sum_{e\in E} \Phi(D(e))\ len(e) \\
\Phi(D(e)) \quad = & \begin{cases}
a_2(e) + b_2(e)\ D(e) + c_2(e)\ D(e)^2 + d_2(e)\ D(e)^3\ , & \text{if } \mathfrak{D}(e,1) \le D(e) \le \mathfrak{D}(e,2)\ ; \\
\vdots & \vdots \\
a_{r_e}(e) + b_{r_e}(e)\ D(e) + c_{r_e}(e)\ D(e)^2 + d_{r_e}(e)\ D(e)^3\ , & \text{if } \mathfrak{D}(e,r_e-1) \le D(e) \le \mathfrak{D}(e,r_e)\ .
\end{cases}
\end{aligned}$$

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`, seem to more easily manage a polynomial with high degree, as compared to r_e different polynomials pieces of degree 3, thus the experiments in §4 use the single polynomial objective function with an algorithmic correction for taking into account the original discrete one (see §4.2). We do note, however, that we believe that the spline approach as considerable potential, but more work would be needed on the side of NLP solvers to realize a computational benefit.

3 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, we smooth the nondifferentiability of the model, and we finally discuss the discrete component of the problem.

3.1 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) employ a "split-pipe model" in which 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 many changes in the diameter along the length of a pipe. Furthermore, there can be additional pressure loss at junctions of the stretches (so called "minor head losses"), which could become significant if there are many different stretches along a pipe; such losses are ignored by all optimization models that we know of. Using this type of split-pipe model, Fujiwara and Khang (1990) 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 2 discrete 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. §4 includes some of these rather negative computational results obtained with the MILP approach.

3.2 Smoothing the nondifferentiability

The main remaining difficulty, besides having given up on global optimality, 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 either get stuck or 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))). But since we ultimately wish to employ available MINLP solvers, and these solvers count on being given smooth NLP subproblems, we take a more promising approach.

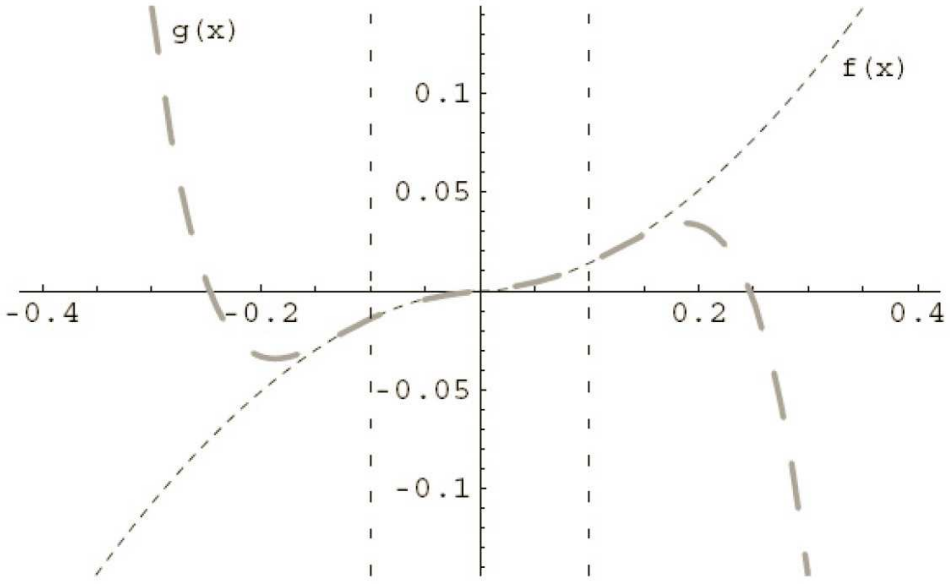
We smooth away the mild nondifferentiability as follows: Let $f(x) = x^p$ ($p = 1.852$) when x is nonnegative, and $f(x) = -f(-x)$ when x is negative (x is standing in for $Q(e)$). This function misbehaves at 0 (the second derivative does not exist there). Choose a small positive δ , and replace f with 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 nice 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 $(-\delta, +\delta)$. Some simple calculations yields:

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

Note that $f'(0) = 0$, while $g'(0)$ is slightly positive.

As can be seen in Figure 1, this seems to work pretty well on a micro level since the function f is not so bad near $x = 0$. In the figure, we have taken $\delta = 0.1$. 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 since 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.

Figure 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).

3.3 Discretizing the diameters

Next, 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, and just passing 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 $X(e,i)$, where $X(e,i) = 1$ 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 that is available in `Bonmin v. trunk`. 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:

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

and

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

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 functions introduced in §2. 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.

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.

3.4 Parameterizing by area rather than diameter

We can use variables:

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

rather than the diameter variables $D(e)$ ($e \in E$). This makes the model "less nonlinear" and "less nonconvex." In particular, we have the *now linear* (previously nonconvex) flow bounds:

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

the *still linear* simple bounds:

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

and the *less nonlinear* and *less nonconvex* head loss across links constraints:

$$H(i) \ - \ H(j) \ = \ \text{sgn}(Q(e))|Q(e)|^{1.852} \ \cdot \ 10.7 \ \cdot \ len(e) \ \cdot \ k(e)^{-1.852} \left(\tfrac{\pi}{4}\right)^{2.435} / A(e)^{2.435} \quad (\forall \ e = (i,j) \in E).$$

Finally, we note that if the cost function is well modeled by a function that is nearly quadratic in diameter, the cost function would be *nearly linear* in area, which could be very nice computationally (note that this is within the realm of possibility; see Figure 3).

We have tried out this area parameterization with different NLP solvers, and it works well, presumably due to the fact that the model is "less nonlinear" and "less nonconvex."

4 Computational experience

In this section we give detailed computational results on instances from both the literature and real-world applications. These results are discussed with respect to previously reported solutions (by sometimes discussing their accuracy) and in light of the use of an open-source MINLP software, namely `Bonmin`. Finally, we also report some (unsatisfactory) computational results on the MILP models obtained with the technique of Artina and Walker (1983), and we highlight why an MINLP approach is in this case far superior to the MILP counterpart.

4.1 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 which 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.

Table 1: Water Networks.

	number of ...					unit
name	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	€

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 (€).

Instances `shamir`, `hanoi`, `blacksburg` and `New York` are taken from the literature, while the others are real-world instances of Italian water networks[1].

For the instances from the literature, the only one that requires some processing of the data in order to fit into our definitions is `New York` which will be separately discussed below. However, the data for 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.

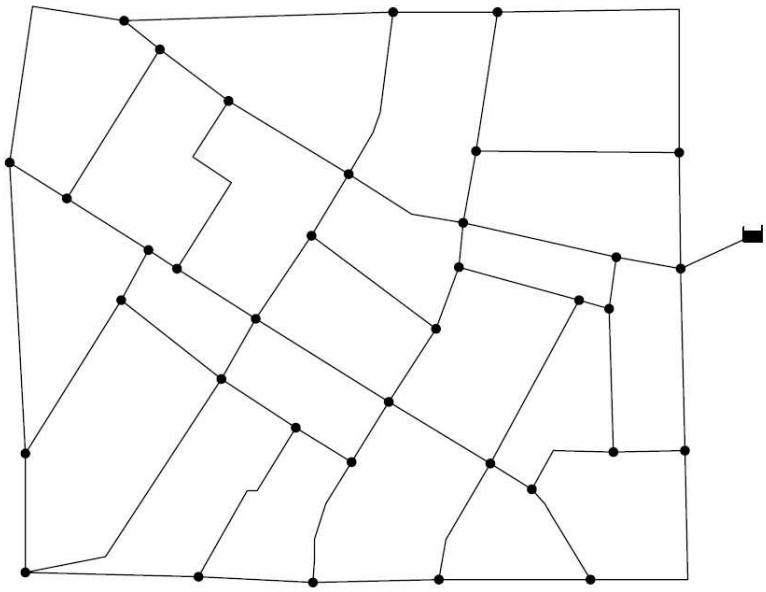
For the real-world instances, the three instances `foss-xyz` refer to a single neighborhood of Bologna, called Fossolo. In Figure 2, we have a diagram of the Fossolo network made with EPANET 2.0 (EPANET (v. 2.0)). EPANET is free software distributed by the US Environmental Protection Agency. It is commonly used to model the hydraulic and water quality behavior of water distribution piping systems.

We have three instances for this network. Instance `foss_poly_0` consists of the original data provided to us and the pipe material for that instance is polyethylene. 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 instance `foss_poly_1` the material for the pipes is again polyethylene but has 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 that of `foss_poly_1` and `foss_iron`, which can be reasonably compared. The value of the solution

[1] All instances are available at `http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`.

Figure 2: Fossolo network.



reported in §4.2 for `foss_poly_1` is much lower than for `foss_iron`. At first this seems surprising, but this is explained by comparing the costs of the varying diameters of pipe. We see in Figure 3 that for small diameters, polyethylene is much cheaper than cast iron, and we note that the data is such that there are feasible solutions with very low flows. Thus, the chosen diameters are in general very small (for example the maximum diameter selected for `foss_iron` is equal to 0.25 meters, for `foss_poly_1` 0.2292 meters).

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 costs, and diameters are up-to-date values in the Italian market.

4.1.1 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.

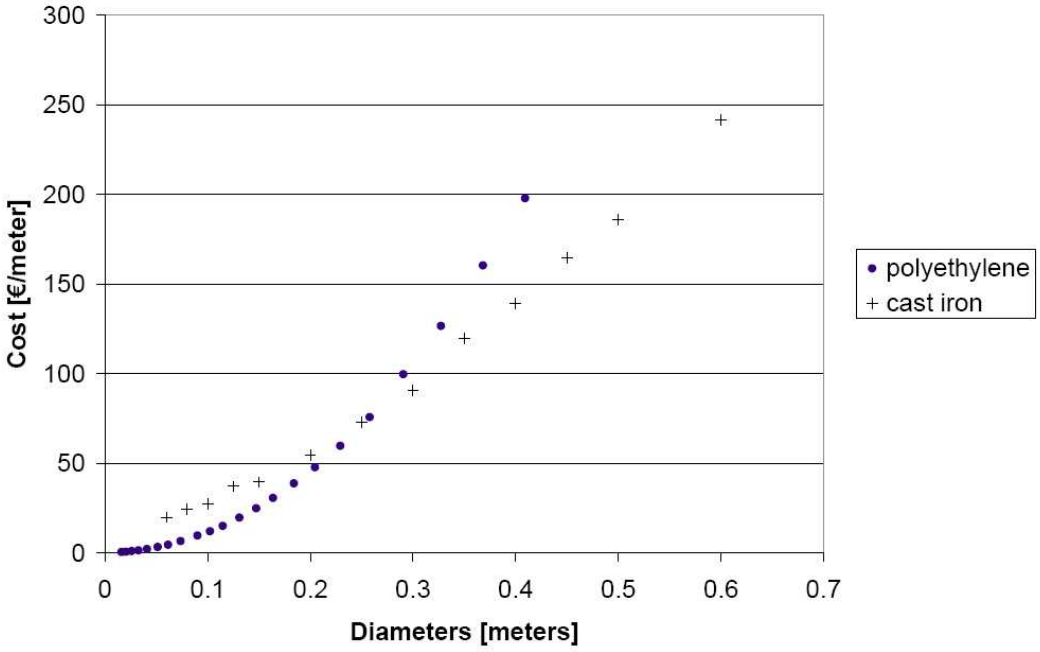
The decisions one has to take are:

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).

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

Figure 3: Cast iron vs. polyethylene.



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

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

whith $r = 0, 1, \ldots, r_e$ 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 equalities 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:

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

Since $len(e)=len(e')=len(e'')$ 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. □

4.2 MINLP results

We have tested our approach using the open-source MINLP solver `Bonmin` (see Bonami et al. (2005), Bonami and Lee (June 2006)). This code was originally developed for finding globally-optimal solutions to MINLPs having convex relaxations, but some accommodations were made to handle nonconvex instances as well, already in the released version `Bonmin v. 0.1`. In fact, some of those accommodations were developed and tested in the context of the present study. Additionally, we made and tested further modifications, to better handle nonconvexities. We implemented these modifications starting from a copy of `Bonmin v. trunk`, i.e., the development version of `Bonmin`. Eventually, some of these modifications may be adopted in a future release of `Bonmin`.

In particular, two main issues came up:

I.1 *Properly evaluating the objective value of integer feasible solutions.* In §2 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 some integer feasible solutions having the approximated objective value worse than the incumbent one but a better value with respect to the correct objective function. Such a behavior has been corrected by allowing `Bonmin v. trunk` to work with *two objective functions*: The first one (i.e., the smooth continuous approximation) is used to guide the search while the second one (i.e., the correct (discrete) objective) is used to evaluate integer feasible solutions, so as to avoid fathoming improving leaves.

I.2 *Heuristically reducing the size of the search space.* The released version `Bonmin v. 0.1` statically defines 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 incumbent. However, we found out that such a static definition of `cutoff_decr` does not fit our purposes since it is hard to define a unique value for all instances. After preliminary computational testing, we implemented in `Bonmin v. trunk` the following policy: The root node continuous value is computed for 50 different starting points (we note that in a nonconvex problem by changing the starting point we end up with different local optima) 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

$$f(\sigma):=\begin{cases}.02\ , & \text{if } \sigma<.1\ ;\\ .05\ , & \text{if } \sigma\geq .1\ .\end{cases}$$

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 reported in Table 2. More precisely, Table 2 reports for each instance the minimum (min), mean (mean) and maximum (max) value of the continuous solution at the root note over the 50 tries. 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 shows that the way we modeled

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

	min	mean	max	std dev	coeff var	# fail	# inf
shamir	401,424.00	413,507.00	673,303.00	37,735.10	0.0912563	0	0
hanoi	5,973,780.00	6,112,600.00	6,241,810.00	88,473.50	0.0144740	0	0
blacksburg	113,417.77	114,534.00	122,648.00	1,659.97	0.0144932	0	0
New York	39,003,936.50	83,480,900.00	112,141,000.00	12,024,900.00	0.1440440	0	0
foss_poly_0	68,638,000.00	78,080,900.00	118,008,000.00	11,096,800.00	0.1421190	0	0
foss_iron	180,600.00	181,977.00	189,632.00	9,081.52	0.0169336	0	0
foss_poly_1	26,640.00	33,076.40	51,494.40	5,470.00	0.1653750	0	0
pescara	1,822,210.00	1,846,930.00	2,153,930.00	66,672.00	0.0360989	0	0
modena	2,565,030.00	2,567,680.00	2,578,490.00	1,920.57	0.0007480	0	0

the problem has a stable behavior in the sense that the continuous solutions have never 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 crucial instances are `New York`, `foss_poly_0` and `foss_poly_1`.

The results obtained using `Bonmin v. trunk` 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 default continuous solution value (NLP), the best continuous solution value among the 50 tries (NLP*), and the value of best MINLP solution with the fitted objective function (MINLP). The last two columns of the table report the value of the best solution found with respect to the true objective function (MINLP$^\circ$) and the value of such a solution mapped on the fitted objective function (fit[MINLP$^\circ$]), respectively. In particular, we marked (with a "$^\circ$") in the last column the three instances for which values MINLP and fit[MINLP$^\circ$] are different, i.e., the instances in which the use of both objective functions simultaneously had a very positive effect.

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

	NLP	NLP*	MINLP	MINLP$^\circ$	fit[MINLP$^\circ$]	
shamir	407,294.57	401,424.00	423,696.31	419,000.00	423,696.31	
hanoi	6,070,967.62	5,973,780.00	6,109,620.78	6,109,620.90	6,109,620.78	
blacksburg	113,417.77	113,417.77	118,251.06	118,251.09	118,251.06	
New York	39,003,936.50	39,003,936.50	39,570,174.42	39,307,799.72	39,784,401.90	$^\circ$
foss_poly_0	79,079,623.25	68,638,000.00	70,842,869.58	70,680,507.90	70,842,869.58	
foss_iron	181,460.80	180,600.00	181,865.00	178,494.14	181,908.30	$^\circ$
foss_poly_1	26,761.80	26,640.00	29,062.82	29,202.99	29,062.82	
pescara	1,822,214.64	1,822,210.00	1,883,480.00	1,837,440.40	1,886,995.93	$^\circ$
modena	2,567,882.92	2,565,030.00	2,620,189.45	2,580,379.53	2,620,189.45	

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 (MINLP$^\circ$) (same column of Table 3), we report the CPU time in seconds to find such a solution (time) and the best MINLP solution value after 1,200 CPU seconds (MINLP^h, 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 three marked in Table 3 with a "$^\circ$" in the last column. Precisely, the effect of modification I.1 above is crucial for the three instances `New York`, `foss_iron` and `pescara` in which the final solution of the fitted objective function is not the best one with respect to the discrete objective function. Without modification I.1 in the code, those improved solutions would have been

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

	MINLP$^{\circ}$	time	MINLP^{h}	# fit	# true
shamir	419,000.00	1	419,000.00	2	2
hanoi	6,109,620.90	357	6,109,620.90	8	8
blacksburg	118,251.09	1,540	118,462.00	6	6
New York	39,307,799.72	3	39,307,799.72	5	6
foss_poly_0	70,680,507.90	1,500	70,721,500.00	6	8
foss_iron	178,494.14	3,070	179,118.11	4	5
foss_poly_1	29,202.99	6,772	29,282.90	6	5
pescara	1,837,440.40	6,701	1,845,650.00	7	21
modena	2,580,379.53	964	2,580,379.53	2	2

rejected. Moreover, Table 4 demonstrates that besides the three above instances, the use of the two objective functions is also effective in the two other `fossolo` instances 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, instead, is an improvement on instances `foss_poly_0` and `foss_poly_1`: Namely, the solution of the former strongly improves from 71,595,394.14 to 70,680,507.90 while the improvement of the latter is smaller (from 29,226.71 to 29,202.99).

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 quite 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 v. trunk` arsenal. In particular, the reported results are all obtained with `tree_search_strategy = dive` and `node_comparison = best-bound` which turned out to be the most stable version. On the other hand, slightly better results on single instances can be obtained with different parameters and in particular using `node_comparison = dynamic`.

The analysis of the best MINLP solutions for the considered instances shows a configuration in which the size of the selected diameters decreases from the reservoir towards the parts of the network farther away from the inlet point. This is depicted in Figure 4 which looks like a tree primary structure without pipes with large diameters isolated in the network[2]. This behavior of the algorithm (not explicitly stated in the model) reflects a general design criterion widely used by practitioners and represents an interesting indicator of the strength of the proposed approach suggesting its practical applicability.

4.2.1 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.

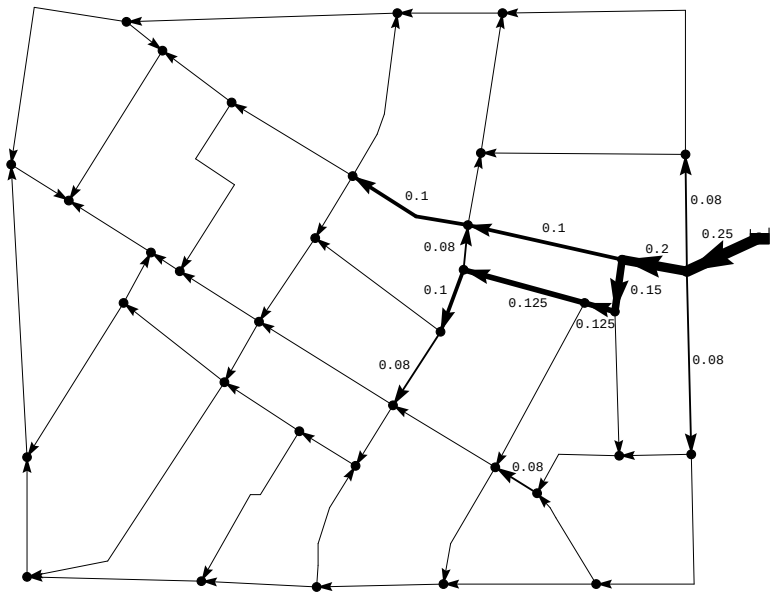
1. For the small instance `shamir`, we find the previously best known (and probably optimal) solution.

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 §4.1, the instance we use is different from the one used in that paper.

3. As stated in §2, 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 a paper to be compared. In particular, in Table 5 we compare our MINLP approach with the approaches in the following papers:

[2]The size of each diameter in Figure 4 is proportional to the thickness of the arc. It is expressed in meters and, for the pipes without the explicit number, it is equal to 0.06, i.e., the minimum diameter of the set.

Figure 4: Solution for Fossolo network, version `foss_iron`.



- 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 5 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 in turn denoted as "MINLP^a" in Table 5. An entry "—" in the table denotes that a particular instance has not been considered in a specific paper.

Table 5: MINLP results compared with literature results.

	Savic and Walters (1997)		Savic and Walters (1997)		Dandy et al. (1996)		Cunha and Sousa (1999)	
	SW99 rel.	MINLP^a	SW99 res.	MINLP^a	DSM96	MINLP^a	CS99	MINLP^a
`hanoi`	6.073 e+06	6.052 e+06	6.195 e+06	6.183 e+06	—	—	6.056 e+06	6.056 e+06
`New York`	37.13 e+06	36.68 e+06	40.42 e+06	40.47 e+06	38.8 e+06	38.8 e+06	—	—

Table 5 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.

4.2.2 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-axes as a function of the head loss $H(i) - H(j)$, on the x-axes. Then, we compute an upper bound on the head loss value as:

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

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:

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

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-axes is the same for every curve obtained by fixing the diameter value. In other words, the x-axes values of the linearization points are common to each diameter, while the corresponding y-axes value changes.

The computational results, obtained by using such an MILP model and 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 inequalities with a standard MILP technique is not a good idea.

Moreover, the MILP approach is heavily dependent on the accuracy of the approximation of the nonlinear component. If such an approximation is highly accurate, i.e., the number of linearization points is large, no MILP feasible solution – a solution which satisfies the constraints of the MILP model – can be found by 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 since 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 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 optimal solution found with 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).

For the `fossolo` set, even with very few linearization points, the MILP approach is unable to find feasible solutions within 2 days. A very inaccurate solution of bad quality has been found for `foss_poly_0` with only 6 linearization points (see Bragalli et al. (2006)). Unfortunately, even providing the MILP model with the set of diameters found by the MINLP approach, no feasible solution can be obtained even allowing 1,000 linearization points.

Finally, for instances `pescara` and `modena`, which are the largest ones, no feasible solutions were obtained, independent of the number of linearization points.

5 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 where known since the 80's 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. This success was 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. The code developed in this context is committed to the `Bonmin v. trunk` repository 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 with significant discrete and nonlinear aspects.

Acknowledgments

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.

References

Alperovits, E., U. Shamir. 1977. Design of optimal water distribution systems. *Water Resources Research* **13** 885–900.

Artina, S., C. Bragalli, A. Lodi, P. Toth. 2002. Approccio MILP al problema di optimal design di reti di distribuzione idrica. *Atti del 28° Congresso Nazionale di Idraulica e Costruzioni Idrauliche (in Italian)*, vol. 1. 67–70.

Artina, S., J. Walker. 1983. Sull'uso della programmazione a valori misti nel dimensionamento di costo minimo di reti in pressione. *Atti dell'Accademia delle Scienze dell'Istituto di Bologna (in Italian)*, vol. Anno 271, Serie III, Tomo X.

Beale, E.M.L., J.A. Tomlin. 1970. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. J. Lawrence, ed., *Proc. of the 5^{th} Int. Conf. on Operations Research*. 447–454.

Bonami, P., L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A. Wächter. 2005. An algorithmic framework for convex mixed integer nonlinear programs. Tech. rep., IBM Research Report RC23771. To appear in *Discete Optimization*.

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

Bonmin. v. 0.1. `projects.coin-or.org/Bonmin`.

Bonmin. v. trunk. `projects.coin-or.org/Bonmin`.

Bragalli, C., C. D'Ambrosio, J. Lee, A. Lodi, P. Toth. 2006. An MINLP solution method for a water network problem. Y. Azar, T. Erlebach, eds., *Algorithms – ESA 2006*, *Lecture Notes in Computer Science*, vol. 4168. Springer–Verlag, Berlin Heidelberg, 696–707.

Coleman, T. F., A. R. Conn. 1982. Nonlinear programming via an exact penalty function: Global analysis. *Mathematical Programming* **24** 137–161.

CPLEX. v. 10.1. `www.ilog.com/products/cplex`.

Cunha, M., J. Sousa. 1999. Water distribution network design optimization: Simulated annealing approach. *J. Water Res. Plan. Manage. Div. Soc. Civ. Eng.* **125** 215–221.

Dandy, G.C., A.R. Simpson, L.J. Murphy. 1996. An improved genetic algorithm for pipe network optimization. *Water Resources Research* **32** 449–458.

Eiger, G., U. Shamir, A. Ben-Tal. 1994. Optimal design of water distribution networks. *Water Resources Research* **30** 2637–2646.

EPANET. v. 2.0. `www.epa.gov/ORD/NRMRL/wswrd/epanet.html`.

Fourer, R., D.M. Gay, B.W. Kernighan. 2003. *AMPL: A Modeling Language for Mathematical Programming*. 2nd ed. Duxbury Press/Brooks/Cole Publishing Co.

Fujiwara, O., D.B. Khang. 1990. A two-phase decomposition method for optimal design of looped water distribution networks. *Water Resources Research* **26** 539–549.

Geem, Z.W. 2005. Optimal cost design of water distribution networks using harmony search. Tech. rep., Environmental Planning and Management, Johns Hopkins University.

Lansey, K.E., L.W. Mays. 1989. Optimization model for water distribution system design. *Journal of Hydraulic Engineering* **115** 1401–1418.

Leyffer, S. April 1998; revised March 1999. User manual for `MINLP_BB`. Tech. rep., University of Dundee.

Liberti, L., C. C. Pantelides. 2003. Convex envelopes of monomials of odd degree. *Journal of Global Optimization* **25** 157–168.

NEOS. v. 5.0. `www-neos.mcs.anl.gov/neos`.

Savic, D. A., G. A. Walters. 1997. Genetic algorithms for the least-cost design of water distribution networks. *ASCE Journal of Water Resources Planning and Management* **123** 67–77.

Schaake, J. C. Jr., D. Lai. 1969. Liner programming and dynamic programming application to water distribution network design. Report 116, Hydrodynamics Laboratory, Department of Civil Engineering, School of Enrineering, Massachusetts Institute of Technology, Cambridge, Massachussets.

Sherali, H. D., E. P. Smith. 1997. A global optimization approach to a water distribution network design problem. *Journal of Global Optimization* **11** 107–132.

Sherali, H. D., S. Subramanian, G. V. Loganathan. 2001. Effective relaxation and partitioning schemes for solving water distribution network design problems to global optimality. *Journal of Global Optimization* **19** 1–26.

Sherali, H. D., R. Totlani, G. V. Loganathan. 1998. Enhanced lower bounds for the global optimization of water distribution networks. *Water Resources Research* **34** 1831–1841.

Walski, T.M. 1984. *Analysis of Water Distribution Systems*. Van Nostrand Reinhold Company, New York, N.Y.

Walski, T.M., D.V. Chase, D.A. Savic. 2001. *Water Distribution Modeling*. Haestad Methods, Inc., Waterbury, CT, U.S.A.

Xu, C., I.C. Goulter. 1999. Reliability-based optimal design of water distribution networks. *Journal of Water Resources Planning and Management* **125** 352–362.