HAL open science

Water network benchmarks for structural analysis algorithms in fault diagnosis

Anna Sztyber, Elodie Chanthery, Louise Travé-Massuyès, Carlos Gustavo Pérez-Zuñiga

To cite this version:

Anna Sztyber, Elodie Chanthery, Louise Travé-Massuyès, Carlos Gustavo Pérez-Zuñiga. Water network benchmarks for structural analysis algorithms in fault diagnosis. 33rd International Workshop on Principle of Diagnosis – DX 2022, LAAS-CNRS-ANITI, Sep 2022, Toulouse, France. hal-03773713

HAL Id: hal-03773713

https://hal.science/hal-03773713v1

Submitted on 9 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Water network benchmarks for structural analysis algorithms in fault diagnosis

A. Sztyber[1] and **E. Chanthery**[2,3] and **L. Travé-Massuyès**[2,3] and **C. G. Pérez-Zuñiga**[4]
[1]Warsaw University of Technology, Warsaw, Poland
[2]LAAS-CNRS, University of Toulouse, CNRS, INSA, Toulouse, France
[3]ANITI, Federal University of Toulouse, Toulouse, France
[4] Engineering Department, Pontifical Catholic University of Peru, PUCP, Peru
e-mail: anna.sztyber@pw.edu.pl, echanthe,louise@laas.fr, gustavo.perez@pucp.pe

Abstract

This paper proposes a set of network benchmarks for diagnostic driven algorithms based on structural analysis. These have been made available in a public repository for use of all the DX community.

1 Introduction

Structural Analysis is a powerful tool in fault diagnosis that abstracts the dynamic equations modeling the system to only the links between equations and variables. Many complex algorithms are developed to work with structural models, especially to generate sets of relations leading to diagnostic tests, i.e. Analytical Redundancy Relations or parity equations, or to select the most relevant ones. However, there is a lack of easily accessible large benchmarks for comparing and testing the algorithm's computational complexity and memory consumption. In this paper, we present a set of structural models of water distribution networks with a wide range of sizes and a Python library for easy and configurable generation of structural models from water network description.

2 Structural models and algorithmic problems

2.1 Structural Analysis

Structural Analysis (SA) is a general framework that can be used to analyze complex, dynamic systems. The main idea is to abstract the dynamic model of a system by just keeping the links between equations and variables. Thanks to this abstraction, SA can be applied to large-scale systems, both linear and non-linear, and even to systems for which the dynamic model and the dynamic parameters are not precisely known. SA relies on a number of very efficient tools related to graph theory [1]. It enables, from a structural model, to find the sets of equations allowing to generate diagnostic tests.

Consider a system described by a set of equations $\Sigma(z, x, \mathtt{f})$ for which z is the vector of known (or measured) variables of a set Z, x is the vector of unknown, i.e., unmeasured variables of a set X and $\mathtt{f}$ is the vector of faults of a set F. Z, X, and F are respectively of cardinal n_z, n_x, and n_f. Each equation is associated to a component of the system. The model generally represents nominal behavior, hence the violation of one equation indicates that the system is faulty and points at the responsible component.

The *structural model* of a system $\Sigma(z, x, \mathtt{f})$ can be obtained by abstracting the functional equations of $\Sigma(z, x, \mathtt{f})$ by structural equations. A structural equation e_i^S abstracts the links between the functional equation e_i and its variables. The structural model can be represented by the *incidence matrix* B, whose rows are associated to equations and columns to variables. $B(i, j) = 1$ indicates that relation e_i includes variables x_j, $B(i, j) = 0$ otherwise.

When used for fault diagnosis, SA may determine subsets of equations that will generate diagnostic tests. The degree of redundancy of a system is defined as the difference between the number of equations and the number of unknown variables included in them. The Structurally Overdetermined (SO) subsets of equations (i.e., with more equations than unknown variables) can be used to build residual generators. Among them, the Minimal Structurally Overdetermined Sets (MSO) are very important. These sets have a structural redundancy of 1 (one more equation than unknown variables). The subset of MSOs with equations impacted by faults (FMSOs) are of particular interest because they allow the generation of a diagnostic test. Some works are also interested in Minimal Test Equation Support (MTES) because they develop more powerful tests.

2.2 Algorithmic problems

Many algorithmic problems are encountered when using structural methods for diagnostic purposes. Among them, a first class of algorithms (**C1**) focus on finding, within a structural model, the subsets of equations representing MTES [2], MSO sets [3] or FMSO sets [4].

Another class of algorithms (**C2**) that can be applied aim at residual generation. Alternative approaches are available and discussed in [5]. Some approaches propose to use data-based method to generate residuals [6].

As many testable subsets of equations can be found in large systems, a third algorithm class (**C3**) solves the problem of selecting MTES or FMSO sets to achieve diagnosability requirements when used for diagnosis. We can refer to FlexDx, proposed by [7]. Another solution is to solve a BILP problem as in [8], or an A^* algorithm [9]. [10] proposes to use a data-driven approach for selecting the best residuals. This class of algorithms can also be used for sensor placement.

3 Structural models of water networks

Water Networks (WN) are a crucial component of each city. It is common to simulate such networks for control and leak detection purposes. EPANET [11] is a popular free simulation software. It is relatively easy to obtain a large variety of network data in EPANET (`.inp`) format.

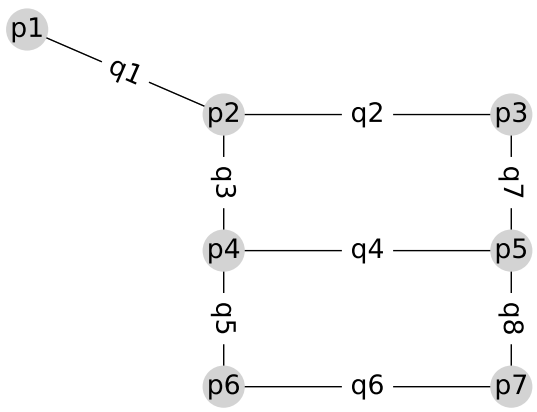


Figure 1: TLN network with variables' names

Water networks can be a useful benchmark for many Fault Detection and Isolation (FDI) algorithms based on SA, because:

- Many networks are available in `.inp` format (small to very large).
- Networks without measurements are just-determined. Adding sensors (pressure or flow), one can get any degree of redundancy from 0 to the number of possible sensors.
- EPANET `.inp` file can be automatically converted to structural model (see 3.3)
- Some networks are highly connected, and complexity grows rapidly with the number of junctions/pipes. Therefore, it is challenging to find all MSO sets, even for small networks.
- Networks can be simulated with EPANET.
- Different types of faults can be injected, the main type being the leak. We can consider leaks in all network junctions or any subset of those. Additionally, sensor faults can be added.

SA methods were already applied to selected water networks for: sensor placement [12; 13] and distributed diagnosis [14].

3.1 Conversion of water network description to structural model

To obtain the structural model of WN, the network structure must be converted to a set of equations. The network structure is given by a graph $G = (V, A)$, where V is the set of vertices corresponding to network nodes and A is the set of edges corresponding to pipes. The set of equations describing the network contains the set of pipe equations, flow balance equations, and measurement equations. The conversion algorithm is based on a method described in [12] with the addition of sensors faults. The network equations are presented following the description in [12].

The example of simple network is shown in Fig. 1 - labels p_i denotes pressure variables (nodes) and labels q_i flow variables (pipes).

Flow balance equations

Each network junction (graph vertex $v \in V$) represents a pressure variable p_j and a flow balance equation:

$$\Sigma_{q_i \in Q_v} q_i = d_v, \tag{1}$$

where Q_v is a set of flows incoming and outgoing to the vertex v and d_v is the demand in node v.

Therefore, for each network vertex v, a structural equation is constructed $e_v^S = \{q_i \in Q_v, f_v\}$, where f_v is a fault variable corresponding to a leak in node v. The demands are assumed to be known and are omitted for simplicity. Consider the node p_2 in Fig. 1. It corresponds to the equation $e_{p_2}^S = \{q_1, q_2, q_3, f_{p_2}\}$.

Pipe equations

Each graph edge ϵ represents a flow variable q_ϵ and the corresponding flow equation:

$$q_\epsilon = sgn(p_i - p_j).c(|p_i - p_j|)^\gamma \tag{2}$$

where q_ϵ is the flow in the pipe corresponding to edge ϵ, p_i and p_j are the pressures of the vertices adjacent to edge $\epsilon = (v_i, v_j)$, and c and γ are parameters modelling physical properties of the pipe.

Therefore, for each network edge ϵ, a structural equation is constructed as $e_\epsilon^S = \{q_\epsilon, p_i, p_j\}$. Consider the edge q_1 in Fig. 1. It corresponds to the structural equation $e_{q_1}^S = \{q_1, p_1, p_2\}$.

Sensors

Each flow (corresponding to edge) and pressure (corresponding to vertex) variable can be measured. For each measured flow variable we construct the structural equation $e_{q_i}^S = \{q_i, m_{q_i}, f_{m_{q_i}}\}$, where q_i is a flow variable, m_{q_i} is a known measurement variable and $f_{m_{q_i}}$ the fault on the sensor. For each measured pressure variable, we construct the measurement equation analogously.

3.2 Networks

As a set of benchmark networks, we consider the networks that were proposed as a benchmark for network design optimization problems [15] [1]. The set of benchmark networks is presented in Fig. 2.

3.3 Implementation

The source code, network input files and generated structural models are provided in a public repository[2].

The program converting EPANET network files (`.inp`) was implemented in Python with packages `wntr` (network loading) and `NetworkX` (graph processing).

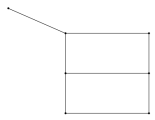
Exemplary `.inp` file for the TLN network (Fig. 1) is shown in Fig. 3. It contains a list of network junctions and pipes with appropriate parameters. It can also contain tanks and pumps and additional information about simulation (units, time of simulation, demand patterns). Some of the networks were slightly modified for cooperation with `wntr` package reading method (no changes in network structure were made).

Network files were read with `wntr` package, converted to `NetworkX` undirected graph and structural model is built from the graph according to the method described in Section 3.1. Model is saved in `json` format as a dictionary with fields:

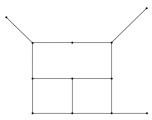
- `model` - dictionary with the equation names as keys and the sets of variables as values

[1] https://emps.exeter.ac.uk/engineering/research/cws/resources/benchmarks/design-resiliance-pareto-fronts/data-files/

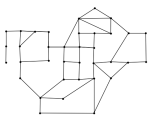
[2] https://github.com/asztyber/wdn-sa-benchmark



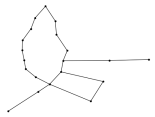
(a) Two-loop Network (TLN)



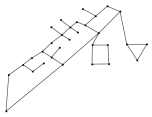
(b) Two-reservoir Network (TRN)



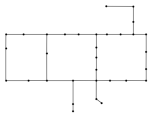
(c) BakRyan Network (BAK)



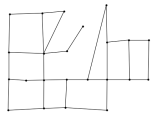
(d) New York Tunnel Network (NYT)



(e) Blacksburg Network (BLA)

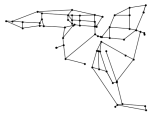


(f) Hanoi Network (HAN)



(g) GoYang Network (GOY)

(h) IP Fossolo Network (FOS)



(i) Pescara Network (PES)

(j) LP Modena Network (MOD)

(k) Balerma Irrigation Network (BIN)

(l) Exeter Network (EXN)

Figure 2: Networks

```
[JUNCTIONS]
;ID Elev        Demand
 2   150         100;
...
 6   165         330;
 7   160         200;
[RESERVOIRS]
;ID               Head
 1                210;
[PIPES]
;ID   Node1   Node2   Length   Diameter
 1    1       2       1000     0.0001;
 2    2       3       1000     0.0001;
...
 8    5       7       1000     0.0001;
```

Figure 3: .inp file for TLN network (dots ... denote shortening for simplicity)

```
{"model":
  {"e0": ["q2", "q1", "q3", "f0"],
   "e1": ["q2", "q7", "f1"],
         ...
   "e6": ["q1", "f6"],
   "e7": ["q2", "p2", "p3"],
         ...
   "e14": ["q6", "p6", "p7"],
   "e15": ["p1", "mp1"],
   "e16": ["p2", "mp2"],
         ...
   "e21": ["p7", "mp7"]},
 "unknown": ["p1", "p2", ..., "q8"],
 "known": ["mp1", "mp2", ..., "mp7"],
 "faults": ["f0", "f1", ..., "f6"]}

# equation name map
{"e0": "ep2", ..., "e7": "eq2", ...,
"e15": "emp1",...}
# fault name map
{"f0": "fp2", ...}
```

Figure 4: Resulting structural model of TLN network (dots ... denote shortening for simplicity)

- `unknown` - list of unknown variables
- `known` - list of known variables
- `faults` - list of faults

The resulting structural model for the TLN network is shown in Fig. 4. The naming convention is as follows:

- equations are labelled with letter `e` and are sequentially numbered from 0,
- flows are labelled with letter `q` and a number; the number corresponds to pipe number in `.inp` file, i.e. `q1` is flow in a pipe with id 1,
- pressures are labelled with letter `p` and a number corresponding to the junction number in `.inp` file, i.e. `p1` is pressure in a junction with id 1,
- faults are labelled with letter `f` and are sequentially numbered from 0,
- measurements are labelled with prefix `m` and variable name.

Additionally, maps of faults and equation names are saved. In the example in Fig. 4, `e0` is the flow balance equation in junction `p2`. It contains fault `f0`, which is a leakage in junction `p2`. `e7` is the equation of pipe `q2`, and `e15` is the measurement equation of the pressure sensor placed in junction `p1`.

The exemplary usage is shown in Fig. 5. `class EpanetConverter` provides the functionality of network conversion to structural model and saves the results in `.json` file. The user should provide to the class constructor: path to the input `.inp` file, list of pressure sensors, list of flow sensors, list of nodes with possible leaks, and boolean information if model should contain sensor faults (`sensor_faults`). The method `structural_from_epanet` converts the network structure to the structural model and the method `save_files` dumps the model and name maps into `json` format. The fields `model`, `eq_name_map` and `f_name_map` contain respectively structural model, equations and faults name maps.

The repository also contains exemplary Python and Matlab usage of the generated structural models.

3.4 Example algorithm application

Created structural models have been analysed with the Python interface of the Fault Diagnosis Toolbox [3] [16] and MSOs and MTES [3] subsets of equations have been calculated. Results for different networks and sensor configurations are shown in Tab. 1. Columns of the table show respectively: network symbol, number of junctions $|V|$, number of pipes $|A|$, number of pressure sensors $|S|$, number of faults $|f|$ (leakages), number of MTES and number of MSOs. In can be noted, that the number of MSOs grows rapidly with the number of sensors (i.e. degree of redundancy) and network size. For large networks, only MTES were calculated because MSO algorithm quickly runs out of RAM (on 8GB machine). It shows that the proposed benchmarks are a challenge to FDI algorithms.

4 Conclusions

The benchmarks available to test algorithms from the DX community have always been few in number, which affects their evaluation. This work exploited an existing water distribution network repository to build a set of benchmarks

[3] https://faultdiagnosistoolbox.github.io/

```
pressure_sensors = ['1', '2', '3']
flow_sensors = ['6']
leaks = ['1', '4', '6']
epn_conv = EpanetConverter(input_file_name, pressure_sensors=pressure_sensors,
                           flow_sensors=flow_sensors, leaks=leaks, sensor_faults=True)
epn_conv.structural_from_epanet()

epn_conv.save_files(output_folder, output_name)
```

Figure 5: Example usage

Name	$\|V\|$	$\|A\|$	$\|S\|$	$\|f\|$	$\|MTES\|$	$\|MSO\|$
TLN	7	8	7	7	7	3934
TRN	12	14	12	12	12	1558096
BAK	36	58	4	36	4035	75861
NYT	20	21	7	20	574	75695
BLA	31	35	4	31	536	3606
HAN	32	34	4	32	289	1840
GOY	23	31	4	23	677	9792
FOS	37	58	4	37	7046	131431
PES	71	99	8	22	151	-
MOD	272	317	4	55	25033	-
BIN	447	454	7	26	611	-
EXN	1893	2467	4	38	2867	-

Table 1: FMSO/MTES results

for diagnostic driven algorithms based on structural analysis. These have been made available in a public repository for use of all the community. Examples have been run both on python and matlab code, showing benchmark flexibility. Tests performed with large networks show that there is still research to be done, at least for scaling up.

Acknowledgments This project is supported by ANITI through the French "Investing for the Future – PIA3" program under the Grant agreement noANR-19-PI3A-0004.

References

[1] J. Cassar and M. Staroswiecki. A structural approach for the design of failure detection and identification systems. In *IFAC Conference on Control of Industrial Systems, vol. 30(6), pp. 841-846*, 1997.

[2] J. Biteus, M. Nyberg, and E. Frisk. An algorithm for computing the diagnoses with minimal cardinality in a distributed system. *Engineering applications of artificial intelligence*, 21(2):269–276, 2008.

[3] M. Krysander, J. Åslund, and M. Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(1):197–206, 2008.

[4] C. G. Pérez, E. Chanthery, L. Travé-Massuyès, and J. Sotomayor. Fault-driven minimal structurally overdetermined set in a distributed context. In *The 27th International Workshop on Principles of Diagnosis: DX-2016*, 2016.

[5] J. Armengol, A. Bregón, T. Escobet, E Gelso, M. Krysander, M. Nyberg, X. Olive, B. Pulido, and L. Travé-Massuyès. Minimal structurally overdetermined sets for residual generation: A comparison of alternative approaches. *IFAC Proceedings Volumes*, 42(8):1480–1485, 2009.

[6] D. Jung. Isolation and localization of unknown faults using neural network-based residuals. *arXiv preprint arXiv:1910.05626*, 2019.

[7] M. Krysander, F. Heintz, J. Roll, and E. Frisk. Flexdx: A reconfigurable diagnosis framework. *Engineering applications of artificial intelligence*, 23(8):1303–1313, 2010.

[8] C.G. Perez-Zuniga, E. Chanthery, L. Travé-Massuyès, and J. Sotomayor. Near-optimal decentralized diagnosis via structural analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[9] E. Chanthery, A. Sztyber, L. Travé-Massuyès, and C. G. Pérez-Zuñiga. Process decomposition and test selection for distributed fault diagnosis. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 914–925. Springer, 2020.

[10] D. Jung and C. Sundstrom. A combined data-driven and model-based residual selection algorithm for fault detection and isolation. *IEEE Transactions on Control Systems Technology*, 27(2):616–630, 2017.

[11] L A. Rossman. *EPANET 2 USERS MANUAL*. U.S. Environmental Protection Agency, Washington, D.C., 2000. EPA/600/R-00/057, 2000.

[12] R. Sarrate, F. Nejjari, and A. Rosich. Sensor placement for fault diagnosis performance maximization in Distribution Networks. *2012 20th Mediterranean Conference on Control and Automation, MED 2012 - Conference Proceedings*, pages 110–115, 2012.

[13] R. Sarrate, J. Blesa, F. Nejjari, and J. Quevedo. Sensor placement for leak detection and location in water distribution networks. *Water Science and Technology: Water Supply*, 14(5):795–803, 2014.

[14] V.Gupta and V. Puig. Decentralized fault diagnosis using analytical redundancy relations: Application to a water distribution network. *2016 European Control Conference, ECC 2016*, pages 1752–1757, 2017.

[15] Q. Wang, M. Guidolin, D. Savic, and Z. Kapelan. Two-Objective Design of Benchmark Problems of a Water Distribution System via MOEAs: Towards the Best-Known Approximation of the True Pareto Front. *J. of Water Resources Planning and Management*, 144, 2015.

[16] E. Frisk, M. Krysander, and D. Jung. A toolbox for analysis and design of model based diagnosis systems for large scale models. *IFAC-PapersOnLine*, 50(1):3287–3293, 2017. 20th IFAC World Congress.