ANL/NSE-20-13

Description of Sensor Assignment Optimization Method as Deployed on a Multi-Node Cluster

Process-Constrained Data Analytics for Sensor Assignment and Calibration

Nuclear Science and Engineering Division

About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free at OSTI.GOV (http://www.osti.gov/), a service of the US Dept. of Energy's Office of Scientific and Technical Information.

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: **orders@ntis.gov**

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728
Email: **reports@osti.gov**

Disclaimer
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

ANL/NSE-20-13

Description of Sensor Assignment Optimization Method as Deployed on a Multi-Node Cluster

Process-Constrained Data Analytics for Sensor Assignment and Calibration

prepared by
Richard B. Vilim, Tim Nguyen, Haoyu Wang and Roberto Ponciroli

Nuclear Science and Engineering Division, Argonne National Laboratory

March 31, 2020

ABSTRACT

Data analytic methods are being developed to address the problem of how to assign a sensor set in a nuclear facility such that a requisite level of process monitoring capability is realized and that the sensor set is sufficiently rich to determine the status of the individual sensors with respect to need for calibration. There is an awareness in the nuclear industry that data analytics combined with rich sensor sets represent a means to improve operations and reduce costs.

In the industry the calibration problem has been previously approached as an empirical data-driven problem with several methods having been developed. However, the experience of the utilities over the past ten years with these methods indicates that the absence of physics-based information renders the data-driven approach less reliable. Complicating factors such as the inherent variability of operation (both equipment alignment and operating condition) can confound a pure data-driven approach while there are no rigorous guidelines for determining what constitutes an adequate sensor set.

The solution under development to overcome these shortcomings supplements the data analytic method with process information in a so-called *process-constrained data-analytic* approach. Simple balance equations are written for generic components (e.g., mechanical pump, valve, and heat exchanger). These do not require *a priori* knowledge of process parameters, such as heat transfer coefficients or friction factors. All that is needed on the part of the utility user is to identify the components and how they are connected.

This report describes the development of a parallel computing capability for determining the optimal sensor set. The optimal sensor set problem suffers from the *curse of dimensionality*. Computation time increases exponentially as the size of the system grows. To overcome this difficulty a pre-conditioner algorithm is developed to find an approximate solution close the actual solution. This serves as a seed for the full-blown algorithm and acts to constrain the space that must searched. The optimization algorithms are described and the implementation on a parallel computing platform is described. The application of the method to a use case we are solving in collaboration with our utility partner served to illustrate how the default sensor set in a nuclear plant may not provide sufficient coverage to infer sensor calibration status.

TABLE OF CONTENTS

ABSTRACT 4
TABLE OF CONTENTS 6
LIST OF FIGURES 7
LIST OF TABLES 8
1 INTRODUCTION 9
2 MONITORING 9
2.1 Physical Plant 10
2.2 Piping and Instrument Diagram 10
2.3 Physics-Based Models 10
2.4 Process Constraints 11
2.5 Quantitative Reasoning 12
3 DEFINITION OF OPTIMIZATION PROBLEM 13
3.1 Sensor Assignment 13
3.2 Cost Function 13
4 OPTIMIZATION ALGORITHM 14
4.1 Mixed Integer Formulation 14
4.2 Genetic Algorithm Solver 14
5 IMPLEMENTATION 15
5.1 Software 15
5.2 Hardware 16
5.3 Cost Function Minimization 17
5.3.1 Rule-based Preconditioner 17
5.3.2 System Code Based 18
6 APPLICATION 18
6.1 Use Case 18
6.2 Piping and Instrument Diagram 18
6.3 Cluster-Based Optimization 19
6.4 Preliminary Results 21
7 CONCLUSIONS 22
8 REFERENCES 24

LIST OF FIGURES

Fig. 1. Schematic of High Pressure Feedwater System heater 10
Fig. 2. Example Piping and Instrumentation Diagram taken from use case 11
Fig. 3. Graphical representation of the adopted procedure to perform the cross-over of selected chromosomes. 14
Fig. 4. Process and Instrumentation Diagram for use case 19
Fig. 5. Select lines from use case P&ID file 20

LIST OF TABLES

Table 1. Diagnostic models for the high-pressure feedwater system 12
Table 2. Fault signatures for single-phase heat exchanger 17
Table 3. List of faults for the high-pressure feedwater system 21
Table 4. Posterior probabilities for the case of a sensor fault at E2.T 22

1 INTRODUCTION

The goal of this research is to develop and demonstrate data-analytic methods to address the problem of how to assign a sensor set in a nuclear facility such that a requisite level of on-line process monitoring capability is realized and that the sensor set is sufficiently rich to determine the status of the individual sensors with respect to calibration. The need for this capability was identified in discussions with utility executives and technical staff at industry workshops and meetings [1][2]. Furthermore, in a recent limited circulation report, the Electric Power Research Institute (EPRI) has identified the use of richer sensor sets in nuclear power plants and data analytics as an innovative means to improve operations and reduce costs.

The assignment of the plant sensor set has in the past been an alarm and control problem: for a given system, what sensor set ensures process condition can be monitored and controlled? In the past, the high cost of sensors and signal transmission with specialized nuclear-grade hardware confined the sensor set to a bare minimum. Recent increase in communication bandwidth and sensor connectivity in hard to reach places due to advances in RF wireless and fiber-optical signal transmission, and sensor price decreases, have lowered the barrier to a richer sensor set at a facility. With these developments, one can now take advantage of new and low-cost hardware options to achieve new and previously unrealizable process and equipment diagnostic capabilities.

While diagnosis of equipment degradation is a sought-after capability, a more urgent need is a reliable method for on-line indication of the calibration status of a sensor. With this information, a utility performing sensor calibration during the refueling shutdown can confine the task to those instruments that require recalibration. Our utility partner has an immediate interest in addressing this problem, while only a longer-term interest in addressing the equipment degradation problem. Moreover, the solution to the latter problem is in part dependent on solving the sensor calibration problem.

This report describes the development of a parallel computing capability for determining the optimal sensor set. The optimal sensor set problem suffers from the *curse of dimensionality*. Computation time increases exponentially as the size of the system grows. To overcome this difficulty a pre-conditioner algorithm is developed to find an approximate solution close to the actual solution. This serves as a seed for the full-blown algorithm and acts to constrain the space that must be searched. The optimization algorithms are described and the implementation on a parallel computing platform is described. The application of the method to a use case we are solving in collaboration with our utility partner is described.

2 MONITORING

The problem of diagnosing equipment performance degradation in thermal-hydraulic systems over the course of plant operation involves collecting and analyzing sensor data on a periodic basis. Given the long time scale over which component degradation typically proceeds, some of the system monitoring sensors may also inevitably degrade and become unreliable. Sensors in nuclear power plants are typically only calibrated once every fuel cycle. The calibration activity requires significant human resources in detecting faulty instruments and recalibrating them. Automated on-line calibration monitoring can be performed during plant operation to detect sensor drifts as they occur and ultimately reduce the O&M cost. It is therefore desirable to have a diagnostic tool with the capability to simultaneously deal with both component faults and sensor faults.

2.1 Physical Plant

In the current state of development, the project is focusing on thermal-fluid systems. The methods under development are general and regard the plant as an arbitrary arrangement of individual components connected through loops and junctions. From a user input standpoint, the plant is viewed as a set of individual equipment components that are connected in a network.

A component that appears in this report for an instance where a plant example is cited is the feedwater heater shown in Fig. 1. In a Rankine cycle this component takes turbine extraction steam to heat feedwater coming from the condenser before it enters the steam generator. The steam side has a condensing region and a subcooled region. On the feedwater side the water is in the subcooled state.

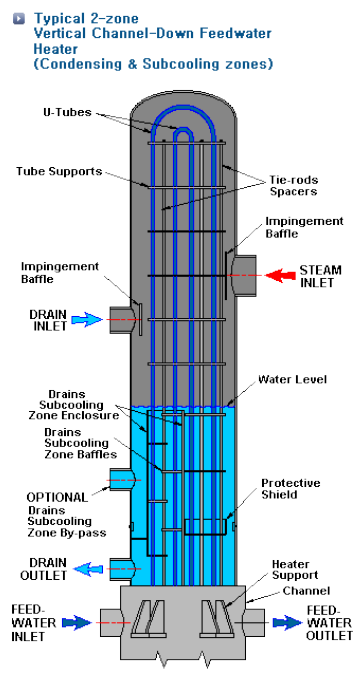

Fig. 1. Schematic of High Pressure Feedwater System heater

2.2 Piping and Instrument Diagram

The physical plant is represented in this project by the Piping and Instrumentation Diagram (P&ID). The P&ID represents a system as an interconnected collection of components and identifies the locations of sensors. The P&ID through the inter-connectedness of components allows the mass, energy, and momentum balances to be written that connect physics-based component models. The P&ID for a portion of the high-pressure feedwater system cited as a use case in this report is shown in Fig. 2.

2.3 Physics-Based Models

A thermal-fluid system in a nuclear power plant can be decomposed into separate components of known generic types, e.g. valve, pump, heat exchanger. Each component of a generic type is designed to perform a basic function of either mass, momentum or energy transfer. The behavior or performance of a component in normal working conditions is described by separate models constructed for each of these three processes. In addition, given the component connections in the P&ID, the conservation equations

for mass, energy, and momentum can also be solved for the network of components, which introduces additional component models, specifically the mixing junction and the splitting tee.

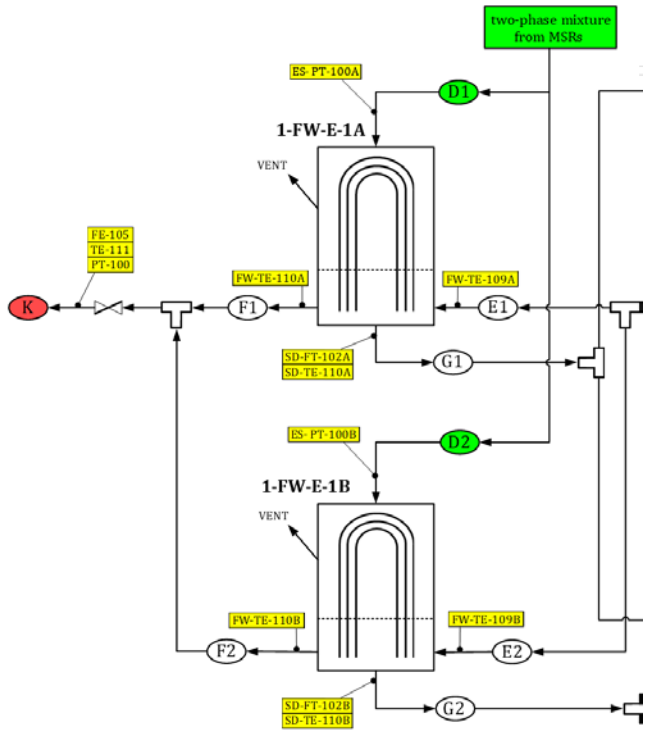

Fig. 2. Example Piping and Instrumentation Diagram taken from use case

Presently the user under this project has available a set of modules for a physics-based modeling of the P&ID. The set includes heat exchangers (single phase and condensing), turbomachines (centrifugal pump), and component connectors (mixing junction and splitting tee). This library of components is accessed in the use case described in Section 6.

A fault is defined to be any change in the characteristics of a component that affect its ability to perform its designed function. Any fault or malfunction in the component would alter its characteristic. Since the models for each component are constructed based on the three conservation equations, any fault in the component would result in an imbalance in at least one of the conservation equations which leads to a non-zero residual for the corresponding model. Presently the project is considering faults of the slow degradation type during which the operation of each component can be considered quasi-static. The faults associated with the P&ID of Fig. 2 are shown in Table 1.

2.4 Process Constraints

In the so-called process-constrained data-analytic approach, simple balance equations are written for generic components as described above under section *2.3 Physics-Based Models* with these equations acting to constrain the model of the process to physical reality. By comparison, there is not explicit provision for ensuring this of a model in a pure data-driven approach.

In this work, these physics-based models are written so that there is not a need to provide *a priori* knowledge of engineering parameters, such as component dimensions and material properties. The model representation allows for these parameters to be determined from measurements taken at a nominal

operating condition. All that is needed on the part of the user is to identify the components and how they are connected, information that is obtained from the P&ID.

Table 1. Diagnostic models for the high-pressure feedwater system

ID	Name	Model Type	Components	Relevant Fault Types
1	DP-1	Generic pressure difference	Feed pumps 1A and 1B, valves, FWH 1A and 1B, pipes	Leakage, Blockage
2	FR-1	Flow ratio	FWHs 1A and 1B, pipes	Leakage, Blockage
3	SDP-1A	Pump head	Drain pump 1A	Pump fault
4	SDP-1B	Pump head	Drain pump 1B	Pump fault
5	SDP-1C	Pump head	Drain pump 1C	Pump fault
6	FWH-1A	HX performance	FWH 1A	Leakage, Fouling
7	FWH-1B	HX performance	FWH 1B	Leakage, Fouling

2.5 Quantitative Reasoning

Once the model equations for each component are calibrated, they can be used as a source of analytical redundancy to check for inconsistencies in the data collected from sensor readings. Residuals, computed as the differences between model predictions and observations from measurement, are evaluated and used to monitor the status of the system. A non-zero residual would serve as an indication that the system has deviated from its normal behavior because of a fault. Since a fault in either the component or any of the sensors involved could result in a non-zero residual, the reasoning process in the diagnostic task will have to distinguish between sensor faults and component faults to be able to localize the fault.

When a residual value change is detected, i.e. a residual is determined to be non-zero statistically, there are three possibilities:

- A component fault: A fault in a component would lead to changes in its behaviors. When the new behaviors are compared to model outputs, which predict the component behaviors under normal working conditions, the changes in component behaviors are reflected in changes of residuals.
- Sensor error: Changes in performance of instruments would lead to change in model outputs, which would also result in changes of the residuals even though the component behaviors remain normal.
- False alarm: With the presence of noise and uncertainty, there is a false alarm rate associated every time a residual is determined to be non-zero by the change detection algorithm.

Component faults and sensor faults can be differentiated based on the evaluation of all the residuals available. Recall that each sensor is used in the calculation of multiple residuals. Each model can be used to compute multiple residuals, each with a different combination of sensors if available. The basic logic, based on which the reasoning rules are formulated, is as follow:

- If a sensor fault occurs, every residual whose calculation involved that sensor would become non-zero.
- If a component fault occurs which is reflected by a component model, every residual computed for that model would be non-zero, regardless of which combination of sensor readings was used.

When the code is initialized, a list of sensors and component models associated with each residual can be generated. Analyzing which residuals are zero and which are not allows the code to pinpoint the fault using an automated reasoning process.

3 DEFINITION OF OPTIMIZATION PROBLEM

3.1 Sensor Assignment

The inverse sensor problem amounts to finding a suitable population of sensors that enable a requisite degree of monitoring capability. More specifically, given a list of faults that need to be detected and isolated to a prescribed degree of spatial resolution, we wish to find the sensor set that will accomplish this goal at the least cost. It is an inverse problem in the sense that it is the opposite of determining what faults can be diagnosed using an existing sensor set. Instead, determine the sensor set needed to provide a requisite fault diagnosis capability.

Through the data analytic approach being developed in this work, the inverse sensor set problem can be solved. To proceed, a set of faults to be diagnosed is drawn up *a priori*. For each component in the system, eligible locations for sensors are tabulated. The inverse sensor algorithm then proceeds to find that subset of possible system sensors which yield the minimum value of the cost function. Algorithmically, for each of the possible subsets, the cost of successfully diagnosing all the faults on the list is computed. Iterating over all sensor subsets yields the subset with the least cost.

3.2 Cost Function

The cost function represents the cost of satisfying a set of diagnostic objectives for a given set of sensors. The cost function in principle should reflect the degree of fault spatial localization achieved, the quality of diagnosis measured in terms of sensitivity to fault severity, and the cost of installing the sensor set. The cost varies with the number and types of sensors in the sensor set. The cost function is minimized to obtain the highest value sensor set. That is, the sensor set that best delivers on the required fault diagnosis capabilities while considering cost.

At this time, we are evaluating a simplified cost function that includes cost of sensor installation and accounts for whether a required diagnosis was in fact made.

$$\text{Total Cost} = \sum_i \sum_j \left(C_{I_j} + C_{S_j} \right) k_{ij} - \sum_m \sum_l \alpha D_{l(m)} \tag{1}$$

where:

$i =$ sensor location on the P&ID,
$j =$ sensor type (e.g. pressure, temperature, …),
$k_{ij} = 1$ for new sensor of type j at location i; $k_{ij} = 0$ otherwise,
$l =$ fault type (e.g. leak, fouling, …),
$m =$ component index,

and:

$C_{I_j} =$ cost of installing sensor of type j,
$C_{S_j} =$ cost of procuring sensor of type j,
$D_{l(m)} =$ diagnosis made for fault of type l in component m,
$\alpha =$ scaling factor.

4 OPTIMIZATION ALGORITHM

4.1 Mixed Integer Formulation

A mixed-integer problem is one where some of the decision variables are constrained to be integer values (i.e. whole numbers such as -1, 0, 1, 2, etc.) at the optimal solution. Typically, in mixed integer problems a binary variable is used to connote a yes/no decision. In our case the decision is whether to install a sensor or not.

It is the binary variable k_{ij} in the cost function of Eq. (1) that renders the sensor assignment optimization problem a mixed integer problem.

Integer variables make an optimization problem non-convex, and difficult to solve. Memory and solution time may rise exponentially as you add more integer variables. When the binary variables number in the few hundreds it may not be computationally feasible to find a solution. In our sensor assignment application, there typically 20 to 30 sensors.

4.2 Genetic Algorithm Solver

hours

	1	2	3	4	5		24
Unit 1	1	1	1	0	0	⋯⋯⋯⋯⋯	0
Unit 2	1	0	1	0	1	⋯⋯⋯⋯⋯	1
⋮							
Unit N1	1	1	0	0	0	⋯⋯⋯⋯⋯	1
⋮							
⋮							
⋮							
Unit N2	1	0	0	1	1	⋯⋯⋯⋯⋯	1
⋮							
Unit NG	1	0	0	0	0	⋯⋯⋯⋯⋯	0

(a)

hours

	1	2	3	4	5		24
Unit 1	0	1	0	0	0	⋯⋯⋯⋯⋯	0
Unit 2	0	0	1	1	1	⋯⋯⋯⋯⋯	1
⋮							
Unit N1	0	0	0	0	1	⋯⋯⋯⋯⋯	1
Unit N1+1	0	1	1	0	0	⋯⋯⋯⋯⋯	1
⋮	0	1	0	0	0	⋯⋯⋯⋯⋯	0
Unit N2-1	1	0	0	0	1	⋯⋯⋯⋯⋯	0
Unit N2	1	1	0	1	1	⋯⋯⋯⋯⋯	1
⋮							
Unit NG	0	0	1	0	0	⋯⋯⋯⋯⋯	0

(b)

Fig. 3. Graphical representation of the adopted procedure to perform the cross-over of selected chromosomes.[3]

The mixed integer problem in this work is approached as a parallelizable problem that makes use of the genetic algorithm solution method. Genetic algorithms are general purpose optimization techniques based on principles inspired from biological evolution as the “survival of the fittest” [4]. The evolution usually

starts from a population of randomly generated individuals (called chromosomes), which represent the candidate solutions of the optimization problem. Each chromosome has a set of properties which can be altered. In the evolutionary process, to establish which individuals should have a higher probability of being allowed to produce offspring (i.e. new problem solutions), and which individuals should have a higher probability of being removed from the population, a suitable, problem-specific metrics (called fitness) needs to be defined [5][6]. According to this metrics, certain chromosomes are stochastically selected from the current population, their genome is recombined and possibly mutated, and form a new generation is formed. Fig. 3 represents a typical cross-over of selected chromosomes to generate a series of offspring.

The "survival of the fittest" principle was applied in the selection of each generation of offspring by discarding the ones yielding high fitness function values. Such evaluation can be accelerated through parallel computing techniques.

5 IMPLEMENTATION

This section describes the implementation on a parallel computing platform for determining the optimal sensor set.

5.1 Software

The Genetic algorithm can be executed in parallel by its nature. The most fundamental parallel computing environment is a multi-core machine. In order to develop and run parallel Python codes on multi-core machines, the software requirements are the same for Windows and Linux machines: One platform and one package.

1. Data Science Platform: "Anaconda 3" [7]. Anaconda is an open-source distribution of Python for data science research. It provides the platform to download and manage 7,500+ Python packages, to develop serial and parallel Python codes, to analyze data with scalability and performance, and to visualize results.
2. Python parallel computing package: "Multiprocessing" [8]. Multiprocessing is an open-source package that supports parallel processing on single multi-core machines either synchronously or asynchronously, which provides the ease and flexibility of parallel computing. The installation and an example are shown in Section 6.1 of this report.

The parallel code can also be executed in a distributed cluster for better performance. In order to run parallel Python codes on a distributed Linux cluster, one additional package is required:

3. Distributed and Parallel Computing package: "Ray" [9]. Ray is a fast and simple framework for building and running distributed applications. It provides a user-friendly framework to create and manage the distributed cluster, and to submit and schedule the computational tasks. The installation and example is shown in Section 6.1 of this report.

Solving the mixed integer problem needs the optimization packages that support integer parameter tuning. Currently there are several candidates available:

1. Mixed Integer Programming Class Library: "MIPCL-PY"[10]. MIPCL-PY is a Python module that facilitates modeling and solving linear and mixed integer programming problems. It supports implementation of both branch-and-cut and branch-and-price algorithms.
2. Python Mixed Integer Linear Programming Library: "Python-MIP"[11]. Python-MIP package provides tools for modeling and solving Mixed-Integer Linear Programming Problems (MIPs) in Python. It includes the COIN-OR Linear Programming Solver - CLP, which is currently the

fastest open source linear programming solver, and the COIN-OR Branch-and-Cut solver - CBC, a highly configurable MIP solver.

The genetic algorithm is the key to accomplishing this research project. In order to perform genetic algorithm optimization, special Python packages are required to provide the GA-based optimization capability in a parallel computing environment. Currently there are two options available:

1. Distributed Evolutionary Algorithms in Python: "DEAP" [12]. DEAP is a novel computation framework including genetic algorithm, evolution strategies, multi-objective optimization and more features. In addition, DEAP supports the parallelization mechanisms such as multiprocessing.
2. Economic Dispatch Genetic AlgoRithm: "EDGAR" [13]. EDGAR is an improved version of a genetic algorithm-based optimization tool focusing on the power generating unit commitment and economic dispatching problem [4]. The EDGAR code was developed in Python by one of the authors at Argonne National Laboratory, and it was deployed in a 24-core Linux server for production purposes.

5.2 Hardware

The computers designed in recent years are largely multi-core machines. The most common configuration has multiple cores and threads in a single CPU package, while some high-performance workstations and servers may have multiple CPU packages installed. Intrinsically, each CPU core is able to execute a set of commands without the assistance of other cores, and such capability gives rise to the opportunity of reducing computational time through parallel computing.

In a properly designed program, the parallel computation on a multi-core machine is executed as follows: One core will serve as the controller and issues the individual computational tasks to multiple worker cores; the worker cores will perform the actual computation independently and return the result to the controller core. Due to the high I/O bandwidth of the communication bus within the same computer, the multi-core machine is capable of yielding a performance nearly proportional to the number of involved cores.

In this project, the Multiprocessing Python package is used to perform the parallel computing on a multi-core machine. It provides the software platform to distribute individual computational tasks to the other physical cores in CPU. Note that the CPU threads (or logical cores) have a relatively limited contribution to the computational power than the CPU physical cores.

The distributed cluster is an expansion to the multi-core computers: it is a computer system composed of multiple computational nodes (individual computers) connected through high-speed network.

Being like the multi-core machine, the cluster usually has one controller node and several worker nodes. The controller node provides users the service of account login, data storage, and computational job submission. The CPU cores in worker nodes perform the actual computation based on the commands and data from the controller node and return the results back to the controller node. The computational tasks are prepared and issued by the job scheduler in the controller node, and the data communication between nodes is done through the network. Due to the limited bandwidth of the network, the communication overhead on top of the actual computational time will be relatively longer than that in a multi-core machine, but the scalability using multiple worker nodes will yield a total performance that usually cannot be achieved by a single multi-core machine.

In this project, the Ray Python package could be used to perform the distributed computing on a Linux cluster. It provides the inter-node communication platform in Python environment, as well as the job

scheduling capability for distributed computing. By performing the computation on multiple nodes simultaneously, the computational time will be significantly reduced.

5.3 Cost Function Minimization

The optimal sensor set problem suffers from the *curse of dimensionality*. Computation time increases exponentially as the size of the system grows. To overcome this difficulty a pre-conditioner algorithm is developed to find an approximate solution close to the actual solution. This serves as a seed for the full-blown algorithm and acts to constrain the space that must be searched.

5.3.1 Rule-based Preconditioner

The computational feasibility is addressed by developing a first approximation to the optimum sensor set for use in initializing the full-blown solution. The assumption is that the default sensor set provided by the system design engineer in the form of the P&ID provides a designer-informed diagnostic capability and thus represents a good starting point. The method begins by identifying the virtual sensors that exist based on inter-component connections given by the P&ID. Given this collection of physical and virtual sensors, the method then identifies all available models for diagnostic purposes and from that determines the structure of the model residuals. A residual-to-fault table can then be generated automatically for the system. The degree to which the list of required faults is diagnosed is assessed.

The diagnostic model is specific to a component and its individual sensor set. For each component type in the P&ID there is a corresponding physics-based model module. Associated with the module are different sensor sets, one of which will match that in the P&ID. And accompanying the module for that particular sensor set is a list of models and possible model residuals.

To see this, consider a counterflow heat exchanger outfitted with sensor set i = 1,2,…6 that denote the sensors for the flow rate, inlet temperature, outlet temperature of the cold side and then those for the hot side, respectively. Consider both component faults and sensor faults:

- Component faults: leakage (denoted by F_L) and fouling (F_0)
- Sensors faults: fault in sensor S_i , denoted by F_i for i = 1, 2,...6

The fault-rule table appearing in the module for that sensor set is shown in Table 2. Each column represents a different set of residual values with the fault giving rise to that particular set of values appearing at the top of the column.

Table 2. Fault signatures for single-phase heat exchanger

	F_L	F_0	F_1	F_2	F_3	F_4	F_5	F_6	*Multiple Faults*	*No Fault*
r_L	1	0	1	1	1	1	1	1	1	0
r_0	1	1	1	1	1	1	1	1	1	0
r_1	1	1	0	1	1	1	1	1	1	0
r_2	1	1	1	0	1	1	1	1	1	0
r_3	1	1	1	1	0	1	1	1	1	0
r_4	1	1	1	1	1	0	1	1	1	0
r_5	1	1	1	1	1	1	0	1	1	0
r_6	1	1	1	1	1	1	1	0	1	0
F_L : leakage, F_0 : fouling, F_i for $i \geq 1$: fault in sensor S_i. 1 *indicates a non-zero residual,* 0 *indicates zero residual*										

The rule-based preconditioner operates by spawning a set of candidate sensors for the P&ID. It then locates for each component in the P&ID the column in the residual-to-fault table that corresponds to the

sensors spawned for that component. From these columns the faults at the top of the column are collected. The cost function is then evaluated using the cost of failing to diagnose a fault and the cost of installing the spawned set of sensors. A new set of sensors is then spawned. This continues until a cost minimum is reached.

5.3.2 System Code Based

With an approximation in the neighborhood of the optimum, the lowest cost sensor set with requisite sensitivity and fault discrimination capability is found using a genetic algorithm (GA) search procedure. The problem formulation includes measurement and model uncertainty. Faulted system scenarios are simulated using a 1-D systems code. A cost function that additionally weighs diagnosis sensitivity is minimized over the set of faulted scenarios.

6 APPLICATION

The application of the method to a use case we are solving in collaboration with our utility partner is described.

6.1 Use Case

The diagnostic framework that has been developed is being demonstrated by application to the high-pressure feedwater system of a PWR plant. The high-pressure feedwater system is part of the condensate and feedwater system that is responsible for the supply of pre-heated feedwater to the steam generators. A detailed description can be found in the final safety analysis report for Unit 1 and 2 of the North Anna Power Station.

6.2 Piping and Instrument Diagram

The P&ID is shown in Fig. 4. Exhaust from the turbines turn into condensate and get heated up by feedwater heaters in multiple heating stages before re-entering the steam generators. The high-pressure feedwater system we are considering here consists of the two heating stages closest to the inlets of the steam generators, referred to as the first-point and second-point stages.

The feedwater heater in each stage is of the closed two-shell type and thus, each heating stage effectively consists of two feedwater heaters in parallel piping lines. Therefore, for this example we have four feedwater heaters: two first-point heaters, labeled by 1-FW-E-1A and 1- FW-E-1B, and two second-point heaters, labeled by 1-FW-E-2A and 1-FW-E-2B, as shown by the P&ID in Fig. 4. The system as shown also include three steam generator feed pumps, labeled by 1-FW-P-1A to -P-1C, and three drain pumps, labeled by 1-SD-P-1A to -P-1C.

From the top of the P&ID, two-phase mixtures from the high-pressure turbine and moisture separator reheaters flow into the shells of the first and second-point heaters. Drains from the first-point heaters flow into the shells of the second-point heaters. On the right of the P&ID, feedwater from later heating stages flows through the two second-point heaters to the suction header of the feed pumps. Drains from the two-point heater shells are collected by the high- pressure heater drain receivers, TK-2A and TK-2B. Drains from the moisture separators (not shown in Fig. 4) are collected by a third drain receiver, TK-2C. The three drain pumps pump condensate from the drain receivers to the suction header of the feed pumps. During normal operation, only two of the three feed pumps operate to pump feedwater through the two first- point heaters to a discharge header to supply the steam generators. In emergency situations, excessive drains from the first-point heaters and the drain receivers are collected by a condenser. All the valves to the condenser are otherwise closed off during normal operation.

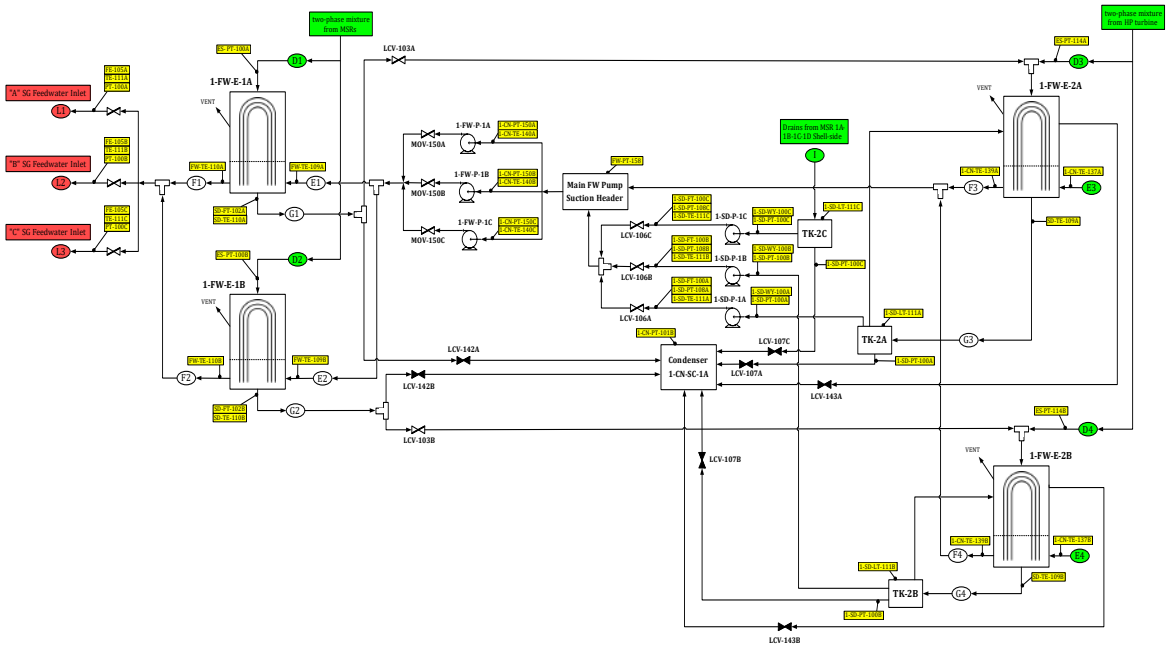

Fig. 4. Process and Instrumentation Diagram for use case

The yellow tags in Fig. 4 indicate the sensors typically available for such system. Each label containing PT denotes a pressure sensor, FE denotes a flowrate sensor and TE denotes a temperature sensor. For brevity, we will use short-handed labels when referring sensors. For example, E2.T, with E2 being the location label and T the variable type, refers to the temperature sensor at the inlet of FWH 1B as shown in Fig. 4. A Dymola simulation model for this system has been developed. Simulation data from the model was used for the analysis in this chapter.

For this demonstration, we excluded the condenser from the P&ID as all its incoming piping lines are normally closed. The components in the system can be characterize by known generic types. Each drain receiver will be treated as coolant tank. Model development for each generic component type is discussed in the remainder of this section.

6.3 Cluster-Based Optimization

This section describes the steps to configure and use the Python parallel computing environment in multi-core machine and multi-node distributed cluster.

The user provides the following case specific information for evaluation of the cost function of Eq. (1).

A. Piping and Instrument Diagram (PID) for use-case plant, which shows components, piping, and default sensor set.
B. Alternate eligible locations for sensors for use case.
C. Cost of sensor and installation for each of pressure, temperature, flowrate, and level measurement.
D. Cost of correct diagnosis but where degree of spatial localization is greater than that of the actual faulted component
E. Cost of failing to diagnose a particular fault.
F. Cost of not uniquely diagnosing a particular fault. i.e. one of the more than one diagnosed faults is correct.
G. Cost of an incorrect diagnosis

The key to the above list is Item A, the P&ID information suitably encoded for input. A select set lines from the input file for the use case is reproduced in Fig. 5.

```
#                                   Input Deck
#` 1. List all components
1, fw_inlet1, system,
2, v_e3, open_valve,
3, fwh_2a_cold, vertical_fwh_cold,
4, junc_1, junction,
5, v01, open_valve,
6, junc_2, junction,

#` 3.  Sensors:
1, fwh_2a_cold, ti, to,
2, fwh_2b_cold, ti, to,
3, fwh_2a_hot, pi, to,
4, fwh_2b_hot, pi, to,
5, sdp_1a, pi, po, to, fo, Wa,
6, sdp_1b, pi, po, to, fo,
7, sdp_1c, pi, po, to, fo,
8, fwp_header, po,

#`  5a. Loops: Components
loop_1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
loop_2, 18, 19, 20, 4,
loop_3, 21, 22, 23, 24, 25,
loop_4, 22, 26, 27, 28,
loop_5, 29, 30, 31, 32, 33, 6,
loop_6, 34, 35, 36, 32,
loop_7, 37, 38, 39, 32,
```

Fig. 5. Select lines from use case P&ID file

To configure and use the “multiprocessing” package on a multi-core machine:

1. Install the Data Science Platform “Anaconda 3”;
2. In Anaconda, install the “multiprocessing” package using the command:
   ```
   ~$ pip install multiprocessing
   ```
3. A simple parallel program is as follows:
   ```
   import multiprocessing as mp

   def func(x):
       return x * x # a function to return the square of a value

   if __name__ == '__main__':
       pool = mp.Pool() # open a parallel pool
       for j in range(1, 100000):
           pool.apply_async(func, args=(j)) # Append the asynchronous jobs to pool
       pool.close()  # Prevents any more tasks from being submitted to the pool
       pool.join()  # Wait for the worker processes to exit.
   ```
4. This program will automatically run on all the available CPU cores in the multi-core machine.

To configure and use the “ray” package on a Linux cluster:

1. Configure the IP address for each node: (the following is an example)
   a. Controller node: 192.168.0.201
   b. Worker node: 192.168.0.202
2. Install the Data Science Platform “Anaconda 3” on all the nodes;
3. In Anaconda, install the “ray” package on all the nodes:
   ```
   ~$ pip install -U ray
   ```

4. On controller node, start the cluster service:
   ```
   ~$ ray start –head –redis-port=1234
   ```
5. On all the worker nodes, join the cluster by indicating the IP address of the controller node:
   ```
   ~$ ray start –address='192.168.0.201:1234'
   ```
6. Now the cluster service is active on all the nodes. A simple distributed program is as follows:
   ```
   import ray

   ray.init(address="192.168.0.201:1234")

   @ray.remote
   def func(x):
       return x * x # a function to return the square of a value

   processes = []
   if __name__ == '__main__':
       for j in range(1, 100000):
           processes.append(func.remote(j)) Append the remote jobs to sequence
   ray.get(processes) Fetch results from the sequence
   ```
7. This program will automatically run on all the available CPU cores in the cluster.

6.4 Preliminary Results

The default sensor set for the high-pressure feedwater system appears in the P&ID of Fig. 4. For each of the components in the P&ID with their respective default sensor set there is an associated residual-fault table in the component model module. The list of realizable faults is shown in Table 3:

Table 3. List of faults for the high-pressure feedwater system

ID	***Comp. Label***	***Comp. Type***	***Fault***
1	1-FW-E-1A	FWH	Fouling
2	1-FW-E-1A	FWH	Tube leak
3	1-FW-E-1A	FWH	Shell leak
4	1-FW-E-1A	FWH	Tube block
5	1-FW-E-1B	FWH	Fouling
6	1-FW-E-1B	FWH	Tube leak
7	1-FW-E-1B	FWH	Shell leak
8	1-FW-E-1B	FWH	Tube block
9	FE-105	Flow sensor	Sensor fault
10	FW-TE-109A	Temp. sensor	Sensor fault
11	FW-TE-110A	Temp. sensor	Sensor fault
12	SD-FT-102A	Flow sensor	Sensor fault
13	SD-TE-110A	Temp. sensor	Sensor fault
14	ES-PT-100A	Press. sensor	Sensor fault
15	FW-TE-109B	Temp. sensor	Sensor fault
16	FW-TE-110B	Temp. sensor	Sensor fault
17	SD-FT-102B	Flow sensor	Sensor fault
18	SD-TE-110B	Temp. sensor	Sensor fault
19	ES-PT-100B	Press. sensor	Sensor fault
20	FW-PT-158	Press. sensor	Sensor fault
21	PT-100	Press. sensor	Sensor fault
22	1-FW-P-1A	Feed pump	Pump fault
23	1-FW-P-1B	Feed pump	Pump fault
24	MOV-150A	Valve	Leakage

25	MOV-150A	Valve	Blockage
26	MOV-150B	Valve	Leakage
27	MOV-150B	Valve	Blockage
28	1-SD-P-1A	Drain pump	Pump fault
29	1-SD-PT-100A	Press. sensor	Sensor fault
30	1-SD-FT-100A	Flow sensor	Sensor fault
31	1-SD-PT-108A	Press. sensor	Sensor fault
32	1-SD-P-1B	Drain pump	Pump fault
33	1-SD-PT-100B	Press. sensor	Sensor fault
34	1-SD-FT-100B	Flow sensor	Sensor fault
35	1-SD-PT-108B	Press. sensor	Sensor fault
36	1-SD-P-1C	Drain pump	Pump fault
37	1-SD-PT-100C	Press. sensor	Sensor fault
38	1-SD-FT-100C	Flow sensor	Sensor fault
39	1-SD-PT-108C	Press. sensor	Sensor fault

In a preliminary demonstration, sensor E2.T (FW-TE-109B) was subjected to a bias that represents a calibration error. The posterior probabilities for the case of this sensor fault are shown in Table 4. The faults with the significant posterior probability are the four sensor faults and FWH 1B shell leakage. One can conclude the sensor faults are more likely than the leakage but, because of the limitation of the current sensor set, cannot differentiate among the four sensor faults.

The inability to uniquely identify sensor E2.T as the failed sensor indicates that the default sensor set does not provide adequate coverage for diagnosing this sensor failure. The sensor assignment problem here then would be to determine that minimum cost sensor addition that would allow for a unique diagnosis to be made. The solution to that problem can be obtained with the methods and algorithms in this report.

Table 4. Posterior probabilities for the case of a sensor fault at E2.T

Fault ID	***Fault***	***Symptoms***	
		r_2 , r_6 , r_9	r_2 , r_6 , r_7 , r_9
15	Sensor E2.T	25.7%	25.8%
16	Sensor F2.T	25.7%	25.8%
17	Sensor G2.w	25.7%	25.8%
18	Sensor G2.T	25.7%	25.8%
5	FWH 1B, Fouling	10.1%	10.1%
7	FWH 1B, Shell leak.	5.1%	5.2%
19	Sensor D2.P	5.0%	5.0%
Other faults		< 0.1%	< 0.1%

7 CONCLUSIONS

The development of a parallel computing capability for determining the optimal sensor set was described. The optimal sensor set problem suffers from the *curse of dimensionality*. Computation time increases exponentially as the size of the system grows. To overcome this difficulty a pre-conditioner algorithm

was developed to find an approximate solution close to the actual solution. This serves as a seed for the full-blown algorithm and acts to constrain the space that must be searched. The optimization algorithms were described and the implementation on a parallel computing platform was described. The application of the method to a use case we are solving in collaboration with our utility partner served to illustrate how the default sensor set in a nuclear plant may not provide sufficient coverage to infer sensor calibration status.

8 REFERENCES

[1] Big Data for Nuclear Power Plants Workshop, The Ohio State University, September 8, 2017.
[2] Nuclear Plant Digitalization, Charlotte, NC, November 13, 2017.
[3] R. Ponciroli, N.E. Stauff, J. Ramsey, F. Ganda and R.B. Vilim, "An Improved Genetic Algorithm approach to the Unit Commitment/Economic Dispatch problem," IEEE Transactions on Power Systems, accepted for publication, March 2020.
[4] D.E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Reading, MA, USA: Addison-Wesley, 1989.
[5] K.S. Swarup and S. Yamashiro, "Unit commitment solution methodology using genetic algorithm", *IEEE Trans. Power Syst.*, vol. 17, pp. 87-91, 2002.
[6] S.A. Kazarlis, A.G. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem", *IEEE Trans. Power Syst.*, vol. 11, pp. 83-92, 1996.
[7] Anaconda 3, https://www.anaconda.com/distribution/
[8] Multiprocessing package, https://docs.python.org/3.7/library/multiprocessing.html
[9] Ray package, https://ray.readthedocs.io/en/latest/
[10] MIPCL-PY package, http://www.mipcl-cpp.appspot.com/mipcl-py.html
[11] Python-MIP package, https://docs.python-mip.com/en/latest/intro.html
[12] DEAP package, https://deap.readthedocs.io/en/master/
[13] Nicolas E. Stauff, R. Ponciroli, T. K. Kim, T. A. Taiwo, "Economic Impact of Flexible Nuclear Operation Estimated with EDGAR Optimization Code," NURER 2018, Sept 30 – Oct 3, 2018, Jeju, Korea.

Nuclear Science and Engineering (NSE) Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC