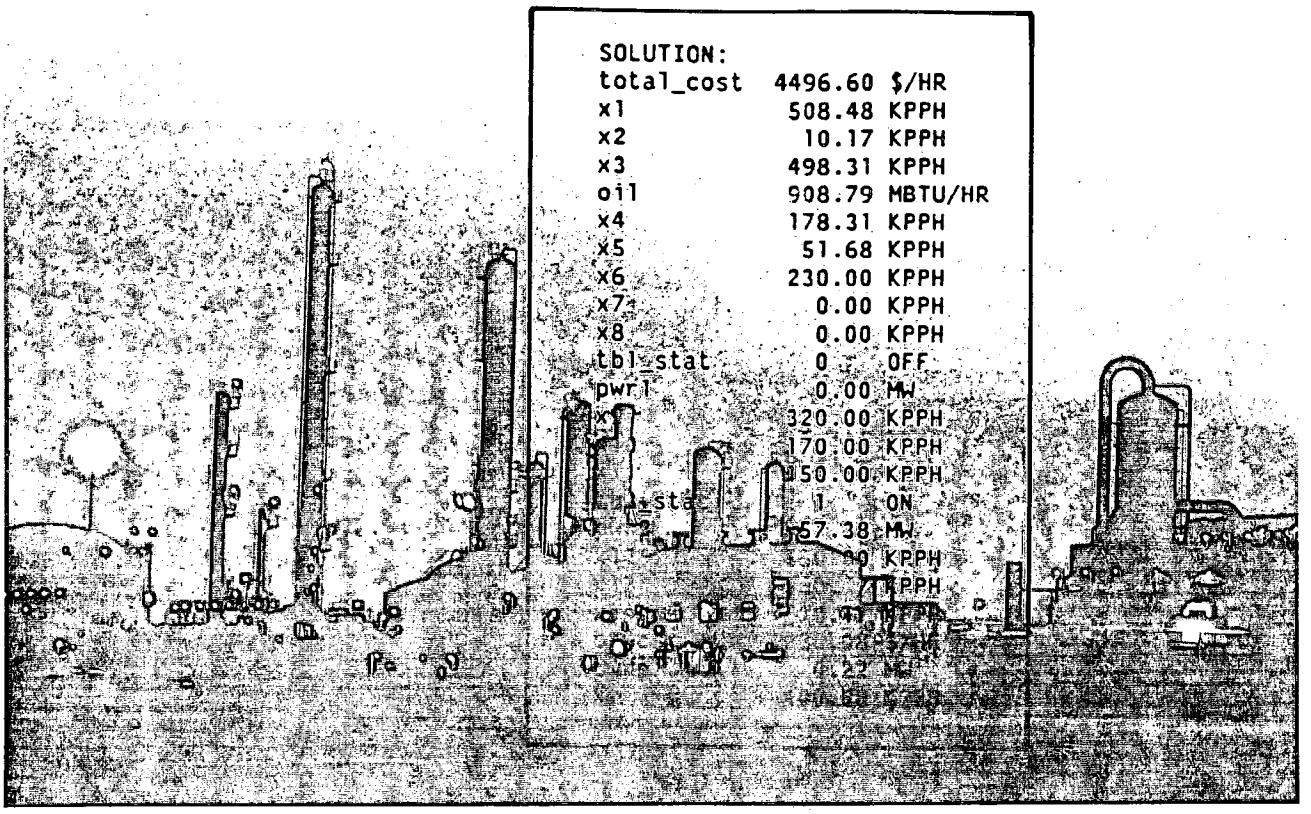
Intelligent Automation Series Optimizer

Cancelled
Per Trans 021-01
May 1997



The optimizer is a production control software package for off-line or on-line steady state optimization. It's modeling language allows you to quickly build a mathematical model of your processes. Fast and reliable mixed integer linear programming algorithms are used for solving optimization problems. These algorithms handle many types of nonlinear constraints and variables of two discrete values.

The optimizer accepts a mathematical model defining objective functions and plant operations as input from the user. It calculates optimum conditions and displays this information to the user.

This program can operate on any linear function and some types of nonlinear functions. It is used frequently to minimize the cost of plant utilities, such as electricity, steam, refrigeration, compressed gas, etc.

The key to successful linear programming optimization is to keep the model as simple as possible to achieve the desired results; however, the model must reflect the actual equipment performance, based on real-time measurements.

FOXBORO®
A SIEBE COMPANY

Changes are easily made to linear programming models to reflect new cost data or equipment removed from service for maintenance. This allows a user to run the optimizer in an off line or "what if" mode to evaluate the effect of future process developments.

The system calculates the set-point values required for the control and I/O software to drive the equipment to their calculated optimum values.

Features of the optimizer include:

- A simple self-contained program language is used to build the process model.
- Algorithms for the process model can be adapted from existing formulas in the optimizer or can be entered as your own equations.
- Optimization results can be observed and evaluated before they are used to direct the process.
- Process units can be switched on or off optimizer control by a single entry in the data base.

Optimizer Structure

The optimizer consists of two separate elements:

- Model Compiler
- Model Optimizer

The user builds a process model and compiles it with the model compiler.

The model optimizer reads the compiled model and builds a linear programming formulation from the data. The model optimizer solves this linear programming problem to determine the optimum value of the variables according to the objective function. These optimum values may be viewed by the user.

When satisfied with the model, the user can link the process measurements to model inputs. Optimizer results can also be linked to process setpoints by the user at this time.

Model Compiler
The user-built process model contains the following information:

- Process Constants: The names and values of all constants in the process model. This information can be displayed and modified at run time by the user.
- Process Variables: The names of all variables used by the optimizer.
- Objective Function: An equation representing the objective of the optimization, such as the minimum total operating cost of a utility.
- Process Equipment: Data identifying each equipment item and every variable related to each equipment item.
- General Constraints: Any user defined algebraic constraints.

The process model is the input to the model compiler; the compiled model is the output.

Model Optimizer
The model optimizer receives the compiled model and converts it into a mixed integer linear program formulation. It forms a matrix of equation coefficients which is manipulated to find the optimum values of independent variables.

The model optimizer may be scheduled to run repeatedly at a specified time interval. Each time the model optimizer runs, it gets the compiled model and gets process measurements. It calculates the optimum values and outputs these values for user evaluation. At the user's command, these data can be linked to the control and I/O software for use as setpoints.

Interface

The optimizer runs on an Intelligent Automation Series Application Processor 10 or 20, or on a personal computer running Intelligent Automation Series software. Relationships between the optimizer and other software are indicated in Figure 1.

Optimizer interface relationships are:

- Keyboard inputs from the user and display outputs for the user occur through the I/A Series Workstation.
- The optimizer can call the physical properties software to calculate values such as enthalpy from steam pressure and temperature.
- Process information such as measurement inputs and setpoint outputs pass between the optimizer and the control and I/O software.

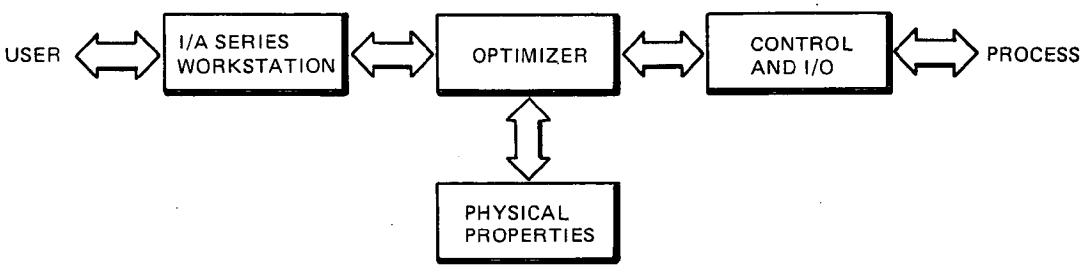


Figure 1. Optimizer Interfaces

Optimizer Operation

Operation of the optimizer is accomplished in six steps:

1. Analyze the process and determine the objective function to be optimized (e.g., total cost). Determine the constraints (e.g., mass, energy, capacity constraints) on the process.
2. Describe the objective function and constraints in the process modeling language provided by the optimizer. The user may take advantage of a library of equipment models provided by the optimizer. For example, the energy balance constraints for a multiple-fuel boiler with nonlinear fuel efficiencies are provided by this library. The process modeling language allows the user to easily describe such a boiler so that the mass and energy constraints are formulated automatically. Equipment models exist for the following units:
   - Boilers (multiple fuel, nonlinear efficiencies)
   - Turbines
   - Pressure Reducing Valves (PRV's)
   - Flash Units
3. Compile the process model using the model compiler.
4. Run the optimizer on this compiled model and look at the results. This amounts to a test of the model in the off line mode. If you are not satisfied with the results, then modify the model and recompile. Otherwise, proceed to step 5.
5. Connect the results to the actual process, through the control and I/O software.
6. Periodically run the process model in the on-line mode to control and optimize the operation.

Example

An example of a simple utility system which can be optimized is shown in Figure 2. This steam power plant provides process steam and electric power to another process. Steam is generated by boiler BLR1, which consumes fuel oil and coal. The efficiency of this boiler is a nonlinear function of the fuel inputs. Two turbines, TB1 and TB2, provide electric power, pwr1 and pwr2 respectively. In addition to these two sources, electric power may be purchased (pwrp) to help satisfy the power demand (pwrd).

The demands for process steam are shown as xp1 and xp2. These demands are satisfied by the turbine extraction streams x8, x10, and the stream x6 from the pressure reducing valve PRV1.

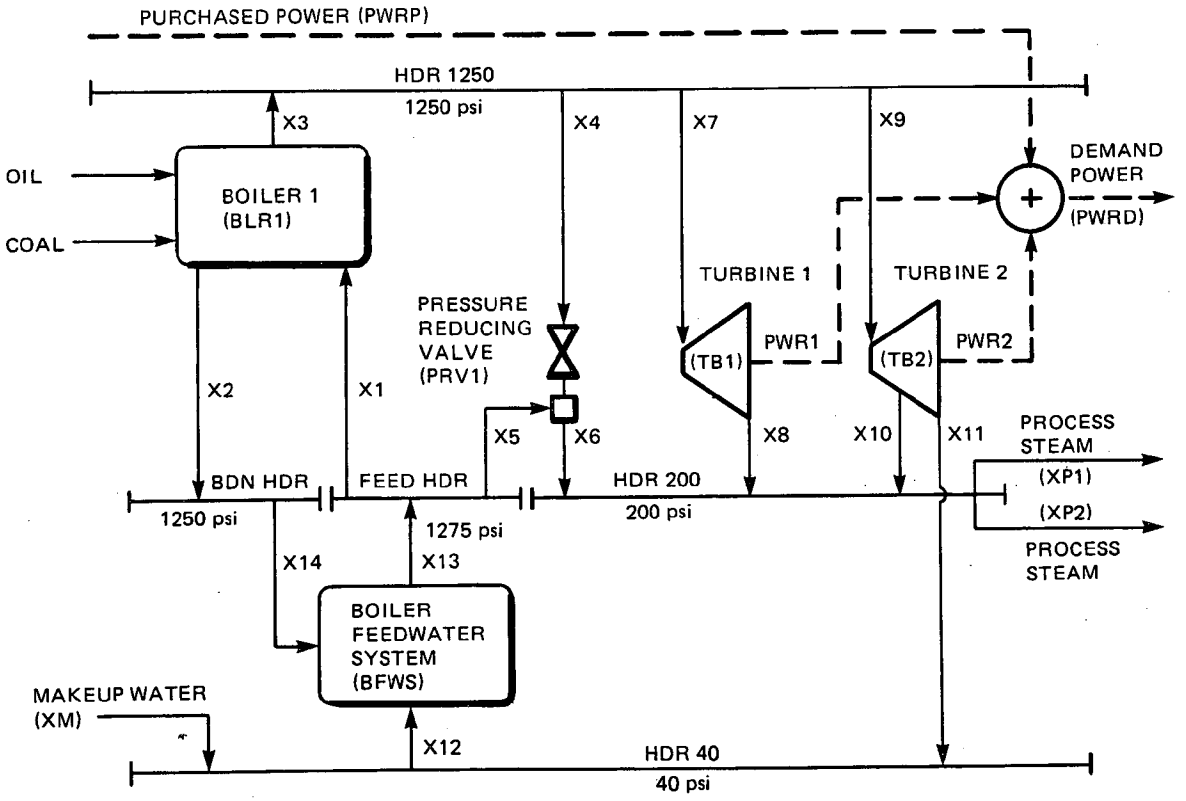


Figure 2. Example of a Utility Process to be Optimized

The objective in optimizing this process is to minimize the costs of the boiler fuels and the purchased power while still satisfying the total demands. A model for this process is easy to create. The objective function can be written as follows:

```
minimize
  /*
   * Minimize fuel costs
   *          + power costs
   *          + feedwater costs.
   */
  total_cost : money = oilcst*oil
                     + coalcst*coal
                     + eleccst*pwrp
                     + bfwcst;
end
```

The mathematical constraints on this process (e.g., mass, energy, capacity) can be expressed as algebraic equations. In addition, constraint equations can be built automatically through the process equipment library. Power demand is an example of a simple algebraic constraint. This constraint is written below. It shows that the power produced by the turbines and the purchased power must satisfy the power demand.

```
/*
 * Power demand constraint
 */
pwr1 + pwr2 + pwrp = pwrd;
```

An example of how the process equipment model allows you to build constraints automatically is given here for the boiler BLR1. A material balance for the boiler is generated automatically from the "in" and "out" statements given here. An energy balance is generated automatically from the energy and efficiency statements.

```
/*
 * Boiler constraints.
 */
boiler BLR1
  in(x1);
  out(x2,x3);
  energy_in( oil, coal); /* fuels = oil, coal */
  efficiency(eff1);
  eff1 = all*oil**2 + a12*oil + b1*coal + c1;
  x2 = 0.02*x1;
  x1 <= 1000.0;
end
```

Using an equipment model from the library does not prevent the user from defining additional algebraic constraints for that equipment. In this example, algebraic constraints are written explicitly for the efficiency (eff1), blowdown (x2) and capacity (x1) of the boiler.

Similarly, the constraints on Turbine 2 can be expressed as follows:

```
/*
 * Turbines constraints.
 */
turbine TB2
  in( x9);
  out( x10, x11);
  energy_out( pwr2);
  x9 = x10 + x11;
  x9 <= 320.0*tb2_stat;
  x11 <= 150.0;
  x11 >= 20.0*tb2 stat;
  pwr2 = 0.0652*x10 + .332*x11 - 3.5*tb2_stat
end
```

This turbine model involves a discrete (0 or 1) variable tb2_stat, representing the status of the turbine (on or off). The optimum status of Turbine 2 is determined by the optimizer.

A model of this entire plant was compiled and optimized for a given set of demands. A partial listing of the results generated by the optimizer are as follows:

CONSTANTS:		
eleccst	60.00	
oilcst	4.50	
coalcst	3.80	
coal	30.00	
xp1	150.00	KPPH
xp2	250.00	KPPH
pwrd	57.60	MW
SOLUTION:		
total_cost	4496.60	$/HR
x1	508.48	KPPH
x2	10.17	KPPH
x3	498.31	KPPH
oil	908.79	MBTU/HR
x4	178.31	KPPH
x5	51.68	KPPH
x6	230.00	KPPH
x7	0.00	KPPH
x8	0.00	KPPH
tb1_stat	0	OFF
pwr1	0.00	MW
x9	320.00	KPPH
x10	170.00	KPPH
x11	150.00	KPPH
tb2_stat	1	ON
pwr2	57.38	MW
x12	550.00	KPPH
x13	560.17	KPPH
x14	10.17	KPPH
bfwcst	280.08	$/HR
pwrp	0.22	MW
xm	400.00	KPPH

These results show that the minimum cost of operating this plant is $4496.60 per hour. It shows that the optimum status of Turbine 1 is OFF and the optimum status of Turbine 2 is ON. There is a small amount of purchased power (pwrp) that could be produced by Turbine 1, but the optimizer has determined that it is cheaper to purchase the power than to run Turbine 1 near its minimum capacity. The optimizer's mixed integer linear programming capabilities allow it to make such a determination.

Much insight into the process economics can be gained from running the optimizer with various changes in the process model. When the user is satisfied that the model accurately represents the real process, these results can be introduced into the control and I/O data base. The model can then be used for on-line supervisory control of the real process.