Abstract—This paper presents an approach for intelligent distributed control of power plants using the concept of multi-agent systems (MAS). Solving the problem of optimally controlling a power plant based on multiple objectives, such as minimizing pollution, maximizing equipment life, etc., and coordinating each of the involved tasks that must be performed in distributed environments is a challenge, which involves many individual computationally intensive tasks. These tasks include calculating feasible control valve operating ranges based on unit load demand, multi-objective optimization, training neural networks, monitoring and managing real-time input/output data, and task delegation, among others. Since each of these tasks requires such computational overhead and these systems need to be coordinated among distributed environments, it is necessary to divide them up into multiple agents. The presented method of design of the multi-agent system is a continuation of research to develop a multi-agent system to implement a technique for computing optimal multi-objective power plant controls.

Index Terms—Power plant control, multi-agent systems, multi-objective optimization, reference governor, model predictive control, automatic gain tuning, intelligent control, distributed control, real-time control, parallel algorithms.

I. INTRODUCTION

Modern power plants are large-scale systems comprised of numerous subsystems, currently controlled by centralized control schemes. Typically, the data processing and operational requirements for such systems are excessive, creating high levels of interconnected logic that are susceptible to system wide failure resulting from a single failure of one component [1]. Moreover, the low-level control systems are incapable of flexibility and autonomy due to the heavy centrality of current control systems. A lack of coordination between locally distributed control systems can cause serious problems due to improper decentralization. With the strained power system grid, it is even more important that power plants have autonomous control. This is not only to provide more efficient power and reduce pollution, but additionally to provide stability to the grid. It is still not uncommon for power plants to experience avoidable accidents that require them to shut down temporarily due to human error [2].

Power plants are complex systems, and the integration of a network of sensor/actuator arrays will make them even more so. They are high in dimension, inherently stochastic, and predominantly composed of a number of interacting subsystems. In networking, data flow is huge, and this situation can be an obstruction for main operation in the power plant. Although with the development in memory technologies and the size of memory is no longer a limitation, control systems have a computational limitation for dealing with huge amount of data. The information processing has been researched in many areas. Further development of tools and methodologies are required to aid performance analysis, measurement of the system, and the evaluation of the organizational and environmental contexts in the systems that have large-scale information. Additionally, autonomous control systems can respond to this data faster, reacting quickly to faults that a human operator would not be able to cope with [2].

The synthesis of the aforementioned features is the prime objective of designing the distributed control system for a power plant. In order to design the distributed control system, an architecture and organizational structure should be established as the foundation of the control system. The task should be proceeded with a systematic approach, and the system needs to have a hierarchical structure for the harmonious operation of the power plant. By utilizing features such as cooperation, negotiation, and competition, control systems can be made more accurate, efficient, and reliable.

In order to satisfy the requirements for the control system, it is proposed to design an intelligent distributed control system based on the multi-agent system (MAS) concept. An agent is a computer software program that is autonomous and situated in the environment. An MAS is a group of many agents which interact with each other.

Goals for implementing the multi-agent system include robustness, adaptability, and modularity. The system is designed to be robust in the sense that it can compensate for the loss of an individual agent at any time. The system is adaptable in the sense that it detects and avoids component failures. Modularity is inherent in the use of a multi-agent system in that the assignable tasks can be modified or added to without changing the functionality of other tasks. This is important because it allows functionality to be added in order to adapt to the changing needs of the system.

The Multi-Agent System based Intelligent Distributed
Control System (MAS-IDCS) can be represented as a synthetic design with intelligence, learning, dynamic organization, and multi-objective optimization based on multi-agent systems. Fig. 1 shows the multi-functional structure of MAS-IDCS. This MAS-IDCS can improve the monitoring in the control system so as to be more efficient, reliable, and serviceable. Furthermore, it is suitable for large-scale systems such as power plants, because it has the unique abilities necessary for solving distributed problems as well as distributed solving of problems by modularity, speedy calculation, reliability, reusability, and knowledge acquisition of the multi-agent system. The developed control system will be demonstrated in a small power plant model.

![Fig. 1. The multi-functional structure of MAS-IDCS.](image)

II. THE POWER PLANT CONTROL SYSTEM

A. The Power Plant Model

The model currently used in the development of the multi-agent system is a 160 MW oil-fired drum-type boiler turbine generator unit. It has been modeled as a third-order three-input three-output multi-input multi-output (MIMO) nonlinear model. The inputs into the system are positions of valve actuators that control the mass flow of fuel (represented as \( u_1 \) in per unit), steam to the turbine (\( u_2 \) in per unit), and feedwater to the drum (\( u_3 \) in per unit). The outputs of the model are electric power (\( E \) in MW), drum steam pressure (\( P \) in kg/cm²), and the drum water level deviation from a set level (\( L \) in m).

The resulting state variables are electric power (\( E \)), drum steam pressure (\( P \)), and steam-water density (\( \rho \)). The dynamic equations for the third-order power plant model were developed by Bell and Åström [3] and are shown below in a summarized form, as given in [2]:

\[
\frac{dP}{dt} = 0.9u_1 - 0.018u_2 P^{9/8} - 0.15u_3 \quad (1a)
\]

\[
\frac{dE}{dt} = ((0.73u_2 - 0.16)P^{9/8} - E) / 10 \quad (1b)
\]

\[
\frac{dP}{dt} = (14u_3 - (1.1u_2 - 0.19)P) / 85 \quad (1c)
\]

The drum water level can be calculated using the equations below:

\[
q_e = (0.85u_2 - 0.14)P + 45.59u_1 - 2.51u_3 - 2.09 \quad (2a)
\]

\[
\alpha_c = (1/ \rho_f - 0.0015) / (1/0.8P - 25.6) - 0.0015 \quad (2b)
\]

\[
L = 50(0.13 \rho_f + 60\alpha_c + 0.11q_e - 65.5) \quad (2c)
\]

where \( \alpha_c \) is the steam quality and \( q_e \) is the evaporation rate in kg/s. The dynamic equations in (1) can be solved for the setpoints \( E_d \) and \( P_d \) in terms of the control inputs by setting the dynamic equations in (1) to zero. These equations are as follows:

\[
E_d = ((0.73u_2 - 0.16) / (0.0018u_2)) (0.9u_1 - 0.15u_3) \quad (3a)
\]

\[
P_d = 14u_3 / (1.1u_2 - 0.19) \quad (3b)
\]

The above equations can be solved to form an inverse static model of the plant, which is useful for generating the feasible ranges for the control values and for performing an optimization. The resulting equations are as follows:

\[
u_1 = (0.0018u_2 P^{9/8} + 0.15u_3) / 0.9 \quad (4a)
\]

\[
u_2 = (0.16P^{9/8} + E) / 0.73P^{9/8} \quad (4b)
\]

\[
u_3 = (1.1u_2 - 0.19)P / 141 \quad (4c)
\]

B. Overall Control Structure

The overall control structure is broken into four main modules: the reference governor, feedforward controller, feedback controller, and the gain optimizer as shown in Fig. 2.

![Fig. 2. Coordinated control structure.](image)
to generate optimal control values \( u_1^*, u_2^*, \) and \( u_3^* \).

Subsequently, the set-point scheduler uses a model, in this case the dynamic equations for the power plant, shown in the next section, to generate the setpoints \( E_d \) and \( P_d \). The setpoints are used as inputs to the feedforward and feedback controllers, as well as the gain optimizer. The gain optimizer generates an optimal set of feedback gains to allow the feedback controller to best control the power plant from one setpoint to another. Using the gains from the gain optimizer, the feedback controller changes the feedforward controls to reach the optimal setpoints more quickly.

Fig. 3. Configuration of the reference governor.

2) The Feedback Controller

The feedback controller consists of three PI control loops, as shown in Fig. 4. This simplification in control can be made since the mass flow of fuel flow, controlled by \( u_1 \), is the largest effecter of output power, \( E \), mass flow of steam to the turbine, \( u_2 \), is the largest effecter of drum steam pressure, \( P \), and the mass flow of feedwater in the drum, \( u_3 \), is the largest effecter of the water level deviation, \( L \). This simplification can be done mathematically by calculating the Relative Gain Array to show input-output interaction and neglecting the elements showing little or no interaction [7].

3) Gain Optimization

Gain optimization is used here to optimize the PI gains used in the feedback controller to more efficiently control the power plant system with respect to the setpoints generated by the reference governor. To do this, hybrid particle swarm optimization (HPSO) is used to find optimal gains for a specific unit load demand curve. HPSO is used as the search algorithm to find optimal gain sets, because it was found to perform better than other methods in the search for optimal steady-state control values [7]. The HPSO algorithm used here for gain optimization is an adaptation of particle swarm optimization (PSO) proposed by [9]. The difference between PSO and HPSO is that in the hybrid method, the worst performing particles are moved to the position of those performing best while keeping their current velocities.

The search space for the optimization was determined by choosing maximum and minimum values for each of the six gains. This creates a six-dimensional search space since there are three proportional gains and three integral gains. The fitness functions for the optimization are to minimize mean-squared error for the errors \( E-E_d \), \( P-P_d \), and \( L-L_d \), and minimize percent overshoot in controlling outputs \( E \), \( P \), and \( L \) with respect to their respective demand levels \( E_d \), \( P_d \), and \( L_d \). Other fitness functions can easily be added to enhance a desired characteristic in the response. Preference values were applied, on \([0,1]\), to these six fitness functions to give more, or less, emphasis to individual fitnesses in the optimization process; 1 indicating most important and 0 signifying omission as a criteria for optimization. For example, if load following is considered more important than meeting pressure and water level requirements, more emphasis can be placed on minimizing the mean-squared error \( E-E_d \) and minimizing overshoot for this control by giving the corresponding preferences higher values than the other mean-squared error and overshoot functions.

III. MULTI-AGENT SYSTEM

A. Overview of Multi-Agent Systems (MAS)

A multi-agent system here is defined as a computer software program that is autonomous and is distributed amongst multiple different environments in order to accomplish a coordinated goal. Furthermore, an agent is said to be intelligent because of its ability to be reactive, proactive, social, flexible, and robust [10]. Each agent adheres in form to an overall architecture, shown in Fig. 5, with the difference between them being task performed and therefore the algorithm modules, scenarios recognized, and plans of action based on the perceived scenario.

Fig. 5. MAS agent architecture.

B. Single Agent Architecture

Individual agents have a common architecture referred to as the agent shell, shown in Fig. 5. The agent shell has two parts, the task thread and the message thread. The task thread executes the algorithms and routines that correspond to an agent’s assigned task. The messaging thread handles incoming
communication from other agents so that the task thread does not have to interrupt what it is doing to receive messages. The message thread will also process messages as much as possible so that the task thread does not have to waste time after the message is relayed. The task thread, however, sends any needed messages to communicate with other agents.

C. Multi-Agent Architecture

The multi-agent architecture defines a three-level hierarchy describing distinct types of functionality an agent can have. The hierarchy consists of high, middle, and low level agents, shown in Fig. 6. The high level, or interface, agents are agents that allow a human operator to interact with, adjust, and monitor parameters and performance in the control system. The middle-level, or managing, agents delegate agent tasks, monitor agent performance, and provide a common location for the agents to acquire, store, or distribute data throughout the system. Low-level agents execute algorithms needed by the control system.

Regardless of functionality, each of these agents is implemented as software running on a high performance computer that is connected to the other agents, and possibly other resources, through an Ethernet connection in a network. The agents proposed for control of the 160 MW power plant are the Interface agent, Delegation agent, Monitoring agent, Database agent, Neural Network agent, the Feedforward agent, the Feedback agent, the Gain Optimizer agent, and the Free agent. Each agent’s software contains all of the code for becoming any of the other agents as they are assigned or sense the need to become [11].

Fig. 6. Three-level hierarchy of agent function structure.

1) Agent Architecture

Each agent, regardless of task, has a common architecture, called the agent shell. The agent shell is composed of a task thread and a messaging thread. These separate threads are achieved using the Matlab Parallel Computing Toolbox, where each thread is run as a separate instance of Matlab. The task thread is the part of the message shell in each agent that executes the algorithms associated with an agent’s assigned task. The functionality an agent assumes is determined by what task it is assigned by the Delegation agent. An agent’s initial task is to be a Free agent, polling the network of agents and standing by for a different task. If an agent is needed, its messaging thread will receive a message from the Delegation agent requesting that it becomes a particular type of agent. At that point, the task thread will begin running the necessary algorithms to complete the task assigned. That agent will continue to run a particular task until completion, or until it is reassigned.

Any data needing to be sent is sent by the task thread to the appropriate agent where it is received by that agent’s messaging thread. The messaging thread is a second instance of Matlab running alongside the messaging task thread. This thread continually listens for messages coming from other agents. Once a message is received, the message thread uses the MAS communication protocol to determine how the message is to be treated based on the agent’s current task. Once the data received by the messaging thread is processed, it is relayed to the task thread for immediate use.

2) MAS Communication Protocol

There are two types of communication necessary within the multi-agent system. The first is the intra-agent communication that takes place between the task and messaging threads. The second is the communication over the network between agents. Intra-agent communication is carried out by synchronizing the memory of the task thread with the messaging thread as new data is received by the messaging thread from other agents. Inter-agent communication is done by sending TCP/IP messages over the network. Network communication is achieved in Matlab using its ability to call Java functions.

3) MAS Message Structure

When an agent sends a message to another agent, it does so using an Agent Communication Language (ACL). The ACL defines the structure of messages being sent. The ACL used here is derived from a protocol proposed by the Foundation for Intelligent Physical Agents (FIPA) [10] to fit the specific needs of this system. An advantage of using such an ACL is that additional messaging capabilities can be added or changed by adding or modifying the necessary fields in the common message structure used by all agents. As shown in Fig. 7, messages in the MAS system contain at least four fields, with an optional fifth field.

![Fig. 7. MAS message field structure.](image)

The first field is the **performatives** field which contains a keyword that informs the receiving agent for what purpose the message is being sent. Possible performatives are request, inform, subscribe, agree, refuse, and notUnderstood. The request-performative informs the messaging thread that its agent will need to either send data or consider becoming another agent. The inform-performative informs the messaging thread that it is being sent data for the task thread. The subscribe-performative informs the messaging thread that a task requested of another agent is going to be performed. The refuse-performative informs the messaging thread that a task requested of another
agent cannot be performed. The notUnderstood-preformative informs the messaging thread that a message sent to another agent was not readable by that agent, meaning either that the agent was not programmed to respond to that type of message or the message was corrupted.

The sender field of the MAS message structure simply contains the identification of the agent sending the message. For example, this field will contain data such as network address and the port that it is sending and receiving messages on.

The receiver field contains the identification of the intended recipient of the message. This field will contain data such as the network address and operating port of the recipient agent. The content field, depending on the type of message, will contain things such as data, the name of a data file, an instruction, or possibly nothing at all. If a request is sent for an agent to take on a certain task, the content field will contain the name of the task. If an inform message is being sent, the content field will contain data or the location of data on the network. If a refuse or notUnderstood message is being sent, the content field may contain a reason for refusal or other error message. If a simple agree message is sent, the content field does not need to contain anything, because the content is already in the performative.

The replyby field is an optional field that is used when a message of importance is being sent and needs to meet a deadline. This field is meant to imply to the recipient agent that this message is of critical importance and needs to take priority over what it is currently doing.

IV. MAS SIMULATION

A. Multi-Agent System Simulation

The agents proposed in [11] for control of the 160 MW power plant model are the Interface agent, Delegation agent, Monitoring agent, Database agent, Neural Network agent, the Feedforward agent, the Feedback agent, the Gain Optimizer agent, and the Free agent. Each agent’s software contains all of the code necessary to become any of the other agents as determined necessary by the system.

To begin a simulation, the power plant model is started along with a number of Free agents, the number depending on the number of agent currently exist, while the power plant model will wait for the initialization signal. Once it has been discovered that a Delegation agent does not exist, the Free agents will coordinate to determine a Delegation agent to monitor and delegate task performance.

Once the Delegation agent is in place, it will begin delegating tasks to the other Free agents, starting with the tasks determined as high priority. The Feedforward and Feedback agents, which contain the reference governor and feedback controller, respectively, are assigned first as they are the core of the control system. At this point, an operator will need to begin an Interface agent, which is the only agent not assigned by the Delegation agent, to send a demand curve for operation of the power plant model.

Once the demand curve is sent to the feedforward agent, the control simulation is started. Upon receipt of the demand curve, the Feedforward agent will begin optimizing steady-state control values and setpoints to send to the Feedback agent. The feedback agent will determine feedback values to compensate the steady-state control values to achieve the setpoints set by the Feedforward agent. When the power plant model receives the first set of control values from the Feedback agent, it will begin its simulation of the power plant and the control simulation has begun. The overall operational MAS control system is shown in Fig. 8.

![Fig. 8. A diagram of an established MAS control system.](image)

It is at this point that the MAS control system is considered operational and other auxiliary agents, such as the Gain Optimizer agent, Neural Network agent, Monitoring agent, and Database agent, can be started. To do this the Delegation agent will assign the remaining tasks, and monitor the other agents to ensure that all of the needed tasks are being performed. If at any point another agent determines that the Delegation agent is no longer operational, depending on its vitality to the control system, it will begin the process of becoming a replacement Delegation agent. Because it would cause undesirable issues in the control system if the Feedforward or Feedback agents suddenly dropped out to become the Delegation agent, they are programmed not to change unless absolutely necessary. The other agents are programmed to respond to this need first, according their ascribed importance.

As the simulation runs, data from the system is saved to a network drive for analysis of the control system. An operator can also monitor the system in real-time, as well as make changes to the system manually. To end the simulation, either a pre-programmed signal built-in to the demand curve is used, or a signal from an operator can be sent to stop the control system and power plant model.

B. Implementing the Gain Optimizer Agent

Once the Delegation agent, Feedforward agent and Feedback agent, have established an operational control system, as in Fig. 8, the Gain Optimizer will be assigned to generate optimal feedback gains in real-time for use in the Feedback agent. To do this, the forecasted unit load demand curve is divided into segments of length \( t_{\text{win}} \), called windows, measured in seconds.

The Feedforward agent will keep track of and send the most imminent unit load demand segment to the Gain Optimizer agent when that load is expected at least \( t_{\text{win}} \) seconds from the
present time. This way the Gain Optimizer agent is always optimizing gains for the next window of expected load, and has a little less than $t_{win}$ seconds to optimize the next set of gains. Since the length of $t_{win}$ determines how much time the Gain Optimizer agent has to optimize, the choice of the length of $t_{win}$ is very important for getting a good result, and will depend on the system being controlled.

Once the Gain Optimizer agent has finished an optimization for the current window of time, it sends the optimized gains to the Feedback agent. Upon receipt, the Feedback agent will store the gains to be implemented when the next window of time begins, which is signalled by the Feedforward agent. This process then continues for the duration of control of the plant. Should the Gain Optimizer fail or something go wrong updating the gains, a predetermined gain schedule should be on hand in the Feedback agent as a backup.

V. RESULTS AND CONCLUSIONS

A. Results

To demonstrate successful operation of the MAS control system, the power plant response to control with and without the MAS was simulated.

The first simulation shows the power plant’s response to control without the MAS architecture, shown in Fig.9. This control system was comprised of the Unit Load Demand Dispatcher, optimal reference governor, PI feedback controller, and power plant model simulator. The system was simulated sequentially as a continuous circuit. The gains used in the feedback controller were determined experimentally such that the system would maintain strict control, but not drive the system unstable at any point of operation. For this simulation, the mean-squared error of the difference between the unit load demand curve for electric power and the plant response was 0.77362.

The second simulation shows the power plant’s response, Fig. 10, to control with the MAS architecture, described previously. The same experimentally determined gains used in the first simulation were used in the Feedback agent in the second simulation. The mean-squared error for the electric power for this simulation was 0.96563. This shows the MAS control scheme to perform comparably to the non-MAS system. However, the mean-squared error is slightly higher for the non-MAS. This is attributed to communication delays between agents.

VI. FURTHER RESEARCH

The next steps in the development of the multi-agent control system will be to substitute neural networks for the power plant model in the reference governor and the gain optimizer [12]. This is important, as most systems do not have models available and yet, a model of the system will have to be used. Part of the benefit of the MAS is that it has the ability to continually adapt the model to be as accurate as possible during online operation. With the implementation of neural networks into the system, another goal will be to develop the Neural Network agent, as the Neural Network agent is crucial to the self-adaptability of the neural network models in the system.

To aid in the development, testing, simulation and validation of the MAS control system, another goal is to look into implementing ideas proposed in papers such as [13,14]. Concepts of interest include synchronizing agents in simulation for the purpose of a fast-as-possible simulation of the control system, and graceful degradation which allows for the control system to maintain control as agents start to fail.

The end goal of this research is to eventually optimally control much larger power plant systems, similar to the
MW system mentioned in [6]. These systems are much more
difficult to model, and will require a similarly complex multi-
agent system in terms of the number of agents needed to
implement an intelligent optimal adaptive control system.

VII. REFERENCES

boiler-turbine units”, IEEE Transactions on Energy Conversion, Vol. 11,
No. 2, pp. 401-406, June 1996.
operations leading up to the blackout of August 14, 2003,” Jan. 12,
2004.
Alternator Units: Data Logs and Parameter Estimation for 160 MW Unit.
control of power plant using particle swarm optimization techniques,”
reference governor for multiobjective optimal power plant operation,”
heuristic optimal control system for a large-scale power plant,” Proc. of
the IEEE World Congress on Computational Intelligence, pp. 5693-
5700, Vancouver, Canada, July 16-21, 2006.
interaction for power plant wide-range operation,” Control Engineering
Practice, 13(12) 1475-1487, 2006.
power plant operation using particle swarm optimization technique,” in
presented at the, IFAC Congr. Prague, Czech Republic, 2005.
IEEE International Conference on Neural Networks, 4, 1942-1948,
1995.
structure specification,” The Foundation for Intelligent Physical Agents,
“Implementation of a multi-agent system for optimized multiobjective
power plant control,” Proc. of the 2010 North American Power
front of a multi-objective optimization problem for a fossil fuel power
plant,” Proc. of the IEEE Power Engineering Society General Meeting,
real-time combined simulation for multi-agent controlled power plants,”
Proc. of the 15th International Conference on Intelligent system
Applications to Power Systems (ISAP 2009), Curitiba, Brazil, November
8-12, 2009.
for Power Plant Control,” Proc. of the 15th International Conference on
Intelligent system Applications to Power Systems (ISAP 2009), Curitiba,
Brazil, November 8-12, 2009.