A Fully Decentralized Approach for Solving the Economic Dispatch Problem

by

Wael Taha Ghareeb ELsayed

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2014

AUTHOR'S DECLARATION

Abstract

A practical formulation of the economic dispatch problem is based on treating the problem as a non-convex optimization problem in which the practical non-convex cost functions are taken into consideration. Formulating the economic dispatch problem as a non-convex optimization problem and finding a better quality solution to this problem has consumed a large portion of the research for decades. Almost all previously presented solutions to the non-convex economic dispatch problem are centralized solutions. Recently, as a result of current research directions towards enabling the smart grid, a new research trend has emerged. This new research trend is to solve the economic dispatch problem using decentralized and distributed mechanisms. Among these mechanisms, the consensus on lambda approach is the best known mechanism. A drawback of this approach is that it can solve only the economic dispatch problem with convex cost functions; in addition, it lacks the appropriate mechanism for incorporating the transmission losses.

This thesis presents a new decentralized approach for solving the economic dispatch problem. The proposed approach consists of either two or three stages. In the first stage, a flooding-based consensus algorithm is proposed in order to achieve consensus among the agents with respect to the units and system data. In the second stage, a suitable algorithm is used for solving the economic dispatch problem locally by each agent. For cases in which a non-deterministic method is used in the second stage, a third stage is applied to achieve consensus on the final solution of the problem, with a flooding-based consensus algorithm for sharing the information required during this stage. The required communication time by the proposed approach has been approximated using JADE software. Four case studies were examined for validation purposes. The results show that the proposed approach is highly effective for both solving the non-convex formulation of the economic dispatch problem and incorporating transmission losses accurately in a fully decentralized manner. Moreover, the proposed approach can also be applied with some adaptation to solve the economic dispatch problem with convex cost functions; in this case, it is very competitive to the consensus on lambda approach.

Acknowledgements

First and foremost, I would like to express my sincere thanks and gratitude to my supervisor, Professor Ehab El-Saadany for his guidance, patience, and support throughout my academic program.

I would like also to thank all the members of my research group: Aboelsood, Ali, Amr, Ayman, Elham, Fatima, Hany, Hassan, Hatem, Maher, Mohamed, Morad, Mostafa, and Omar who contributed so wonderfully to my learning experience.

I would like to offer my sincere thanks and gratitude to Prof. Yasser G. Hegazy for enlightening this path to me.

Finally, I would like to thank my parents, and my brothers for their encouragement and continuous support.

Dedication

To my dear parents, and my brothers.

To my professors and supervisors who taught me during my undergraduate and graduate studies.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Preface

Solving the Economic Dispatch Problem (EDP) is one of the most important tools of power system operation. The objective of solving it is to minimize the total generation cost of the generation units while satisfying numerous constraints associated with both the units and the system. The simplest formulation of the EDP involves convex cost functions and neglects many practical cost functions, such as those involving the valve-point effect or prohibited operating zones and those for multiple fuel units. The simplest formulation is either an approximated formulation or, when many practical non-convex cost functions are present in the actual system, a completely improper formulation. Such an improper formulation and the solution based on it may result in monetary losses in the order of millions of dollars per year. A more practical EDP formulation is based on treating the problem as a non-convex optimization problem in which practical non-convex cost functions are taken into consideration. Almost all previously presented solutions to the non-convex economic dispatch problem (NCEDP) are centralized solutions wherein the problem is solved by a central authority. Recent studies directed at enabling smart grids have led to a new research trend: the development and investigation of solutions to the EDP based on decentralized and distributed mechanisms [1]-[6]. The primary motivations for this trend are as follows:

- The extensive employment of smart grid concepts will lead to communication congestion and complexity in central management systems. The complexity inherent in centralized controllers may make it difficult for system operators to act on information collected from smart grid sensors in an appropriate time frame [1], and the resultant communication congestion requires a high-bandwidth communication infrastructure [5].
- Fully decentralized systems do not give rise to concerns about reliability issues related to single point failure [1]-[6].
- Distributed and decentralized systems are more scalable and more flexible with respect to system changes than centralized systems and hence can effectively

accommodate the variable topology and the plug-and-play feature associated with smart grids [4].

Based on these factors, the need for consideration and investigation of decentralized EDP solutions is apparent. However, the literature describes only a few attempts to solve the NCEDP in a distributed manner. Almost all previously proposed distributed and decentralized algorithms are based on an approach involving consensus on the incremental cost variable [1]-[4]. This approach can be used only for solving the EDP with convex cost functions in a distributed manner, and transmission losses cannot be appropriately or accurately incorporated. Previous attempts to incorporate transmission losses, such as in [1] and [5], are based on the assumption that the loss coefficients are always constant or are provided to the agents by a central authority. Another attempt to incorporate transmission losses into a distributed algorithm is that proposed in [4]; however, the results produced by the first case study reported in [4] show that the methodology used to incorporate the transmission losses yields inaccurate results, especially when a substantial change in system state occurs between one dispatching period and the next, such as a significant change in system load. The only attempt to solve the NCEDP using a distributed algorithm is that reported in [5]; nevertheless, in addition to the disadvantage of the assumption of constant loss coefficients or central authority assistance in computing the loss coefficients, another inadequacy is that better-quality solutions for the NCEDP can be obtained if an efficient metaheuristic technique are used rather than the deterministic algorithm proposed by the authors in [5]. The advantages of the approach presented in this thesis are as follows:

- The proposed approach is fully decentralized, with no need of a central authority to compute the total number of agents, the total system load, or the transmission losses.
- The proposed approach can be adapted for solving both the convex and the practical non-convex formulation of the EDP.
- Transmission losses are incorporated effectively.
- Because the proposed approach can share transmission line data and bus data among the agents in a reasonable timespan that is equal to or less than a few seconds, the proposed fully decentralized approach can be adapted for solving other power system optimization problems in a fully decentralized manner: security-constrained economic dispatch, unit commitment, and optimal power flow.

1.2 Objectives

The main objectives of this thesis are as follows:

- Studying and investigating previously proposed decentralized and distributed approaches for solving the economic dispatch problem.
- Developing a new decentralized approach which heals the limitations of the previously proposed approaches for solving the problem.
- Providing the suitable case studies and experimentation for validating the operation of the proposed approach and for comparing it with previously proposed approaches.

1.3 Thesis Outline

The thesis consists of five chapters. In chapter 2, a general overview about the EDP and the distributed algorithms is presented followed by a complete literature survey of the previous decentralized and distributed solutions to the EDP. Chapter 3 introduces the proposed approach for solving the EDP in a fully decentralized manner. The case studies are provided in Chapter 4. Finally, discussion and conclusion are presented in Chapter 5.

Chapter 2
Literature Survey

The purpose of this chapter is to present a literature survey of the decentralized and distributed solutions to the economic dispatch problem. Before presenting this survey, some important definitions are reviewed.

2.1 Formulation of the Economic Dispatch Problem

2.1.1 Simple Formulation

The following represents a simple formulation of the EDP:

$$\text{Minimize} \quad C_T = \sum_{i=1}^{n} C_i(P_i) \tag{2.1}$$

Subject to

$$P_i^{\min} \le P_i \le P_i^{\max} \tag{2.2}$$

$$\sum_{i=1}^{n} P_i = P_D + P_L \tag{2.3}$$

Where $C_i(P_i)$ is the cost function of generation unit i; P_i is the power output of generation unit i; n is the number of generators; P_D is the total system load; $P_i^{\min}$, $P_i^{\max}$ are the lower and upper limits of generation unit i, respectively; and P_L is the total system losses computed using Kron's loss formula, as follows:

$$P_L = \sum_{i=1}^{n} \sum_{j=1}^{n} P_i B_{ij} P_j + \sum_{i=1}^{n} B_{0i} P_i + B_{00} \tag{2.4}$$

Where B, B_0, B_{00} are the loss coefficients. The generation cost function is modelled with the following quadratic formula:

$$C_i(P_i) = a_i + b_i P_i + c_i P_i^2 \tag{2.5}$$

Where a_i, b_i, and c_i are the fuel cost coefficients of unit i. In the above formulation, when the transmission line losses are neglected, this formulation becomes the simplest formulation to the EDP.

2.1.2 EDP Considering the Valve-Point Effects

Real input-output cost curves of generator units are non-convex due to valve-point effects. Cost functions that include the valve-point effects can be written as follows [7]

$$C_i(P_{Gi}) = a_i + b_i P_{Gi} + c_i P_{Gi}^2 + d_i \times \left|\sin(e_i \times (P_{Gi}^{\min} - P_{Gi}))\right| \tag{2.6}$$

Where d_i and e_i are cost coefficients of unit i. Figure 2.1 shows two cost curves; one with valve-point effects and the second without valve-point effects.

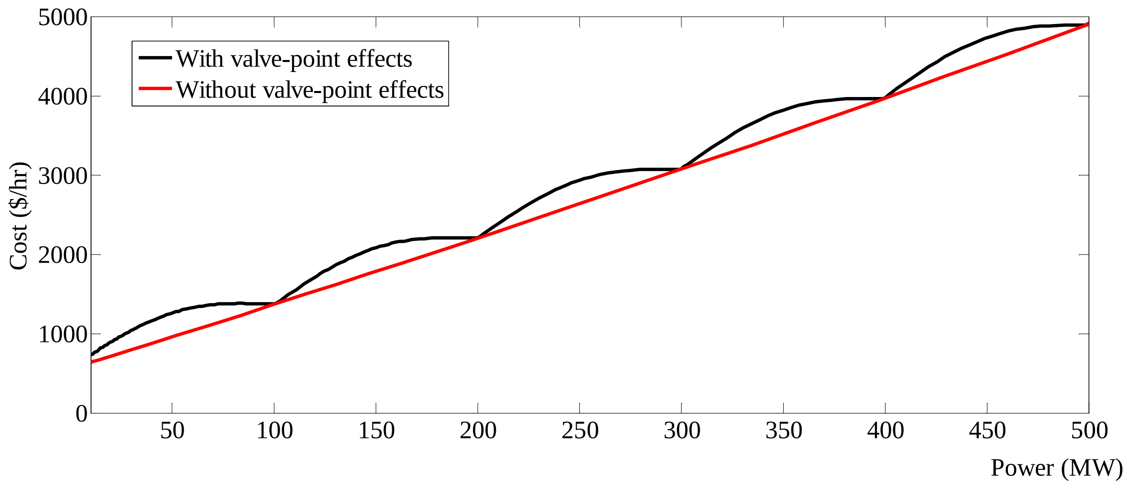


Figure 2.1 Generation unit cost curves with and without valve-point effects

2.1.3 EDP Considering Multiple Fuels Units

Practically, some thermal generation units are supplied with multiple fuels such as oil and natural gas. This requires that each unit be modelled with several piecewise quadratic functions as follows:

$$C_i(P_{Gi}) = \begin{cases} a_{i1} + b_{i1} P_{Gi} + c_{i1} P_{Gi}^2, & fuel\ 1,\ P_{Gi}^{\min} \le P_{Gi} \le P_{Gi1} \\ a_{i2} + b_{i2} P_{Gi} + c_{i2} P_{Gi}^2, & fuel\ 2,\ P_{Gi1} < P_{Gi} \le P_{Gi2} \\ \vdots \quad \vdots \quad \vdots & \vdots \quad \vdots \\ a_{in} + b_{in} P_{Gi} + c_{in} P_{Gi}^2, & fuel\ n,\ P_{Gin-1} < P_{Gi} \le P_{Gi}^{\max} \end{cases} \tag{2.7}$$

Where a_{in},b_{in},c_{in} are cost function coefficients of unit i with fuel type n. Figure 2.2 shows the effect of considering multiple fuels on the cost function shape of a certain unit.

2.1.4 EDP Considering Prohibited Operating Zones

Physical operating limitations may result in cost curves with prohibited operating zones. To model these zones, the following constraints must be added to the problem formulation:

$$\begin{aligned} &P_i^{\min} \le P_i \le P_{i,1}^{l} && (i = 1, 2, ..., \mathrm{n}) \\ &P_{i,j-1}^{u} \le P_i \le P_{i,j}^{l} && (j = 2, 3, ..., nj)\ (i = 1, 2, ..., \mathrm{n}) \\ &P_{i,nj}^{u} \le P_i \le P_i^{\max} && (i = 1, 2, ..., \mathrm{n}) \end{aligned} \tag{2.8}$$

Where $P_{i,j}^{l}$ is the lower bound of the jth prohibited operating zone of unit i, $P_{i,j}^{u}$ is the upper bound of the jth prohibited operating zone of unit i, and nj is the total number of prohibited operating zones in unit i. Figure 2.3 shows an example of cost function with prohibited operating zones.

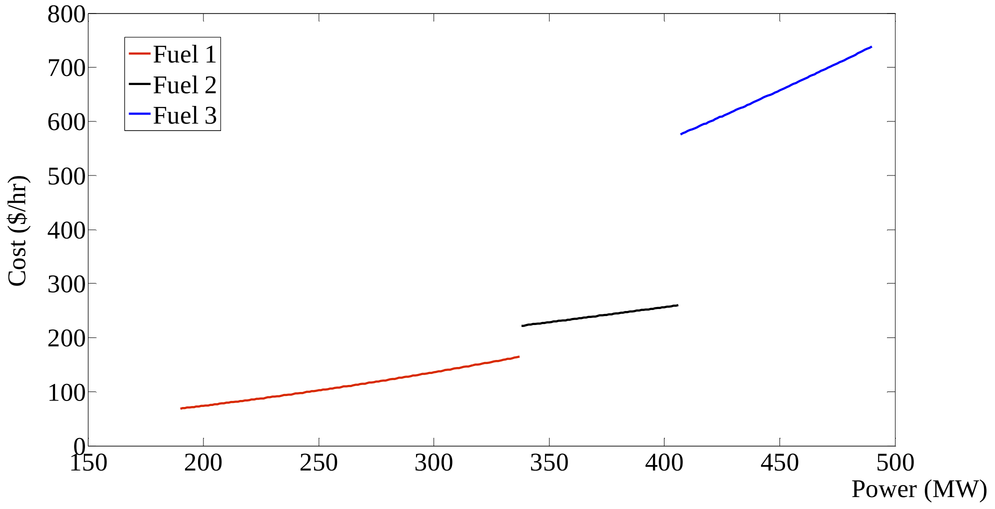


Figure 2.2 Example of cost curve for multiple fuels unit

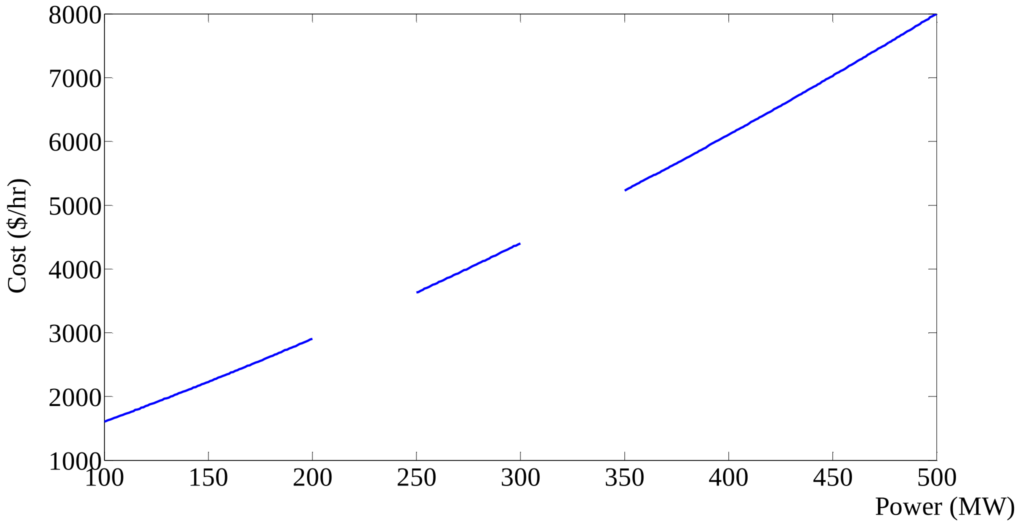


Figure 2.3 Example of cost curve with prohibited operating zones

2.2 Multi-agent Systems

2.2.1 An agent

An agent can be defined as a software component that has some specific characteristics. The most important characteristics of agents are [8]:

- Reactivity: agents can perceive the surrounding environment through sensors and measuring devices and can respond to changes in the environment.
- Proactivity: agents can not only take actions in response to changes in the environment but can also initiate actions to achieve predefined goals at specific time points.
- Autonomy: agents can take decisions like changing their internal state or interacting with the surrounding environment without human intervention.
- Sociality: agents can communicate and cooperate with humans and/or other agents to achieve certain goals.

2.2.2 Multi-agent System

Multi-agent system is a group of agents that are able to interact with the environment. The Multi-agent system can solve difficult problems that cannot be solved by an individual agent like managing complex systems.

2.2.3 The FIBA Agent Communication Language

For the agents to communicate with each other, communication languages are needed. In 1995, FIPA was established. FIPA stands for Foundation for Intelligent, Physical Agents. It consists of a collection of academic and industrial organizations. FIPA' main goal is to develop a set of standards and agent communication languages for software agent technologies. FIPA developed the FIBA Agent Communication Language (FIPA-ACL). FIPA-ACL is based on the speech act theory which states that each message has an intention or communication act. For this reason, each message has an accompanying performative or act beside the message content. This performative specifies the intention of sending the message and what the sender expects the receiver to do with the content of the message. Some examples of performatives are inform, accept, refuse, propose, not-understand or confirm.

2.2.4 Ontology

The agents have to agree about a certain set of terminologies and concepts in order to understand each other. This set of terminologies and concepts constitutes a body of knowledge that is known as the ontology. For example, in power systems, if an agent sends a message to another agent asking about “the power generated”, the second agent should be able to understand what is meant by the terminology “the power generated”. Another example, if an agent sends a message to another agent that controls a generating unit asking to “increase the output power of the unit”, the second agent should be able to understand what is meant by the concept “increasing the output power”. Defining a set of terminologies and concepts for the agents so that they can understand each other means defining ontology for them.

2.2.5 Protocols

Protocols are the rules that govern and organize the communication between the agents. Defining a set of rules that organize the interaction and communication between the agents means defining some protocols.

2.2.6 Distributed Systems

A distributed system is a collection of distributed entities that are connected with a network. In distributed systems, the communication between the distributed entities is usually done while processing the data and while taking decisions or solving problems.

2.2.7 Fully Distributed Systems

Fully distributed systems are distributed systems in which there is no leader or commander for organizing and coordinating the interaction between the agents. In fully distributed systems, the agents do not need a centralized authority or a leader in order to accomplish their tasks and achieve their goals successfully.

2.2.8 Decentralized Systems

In decentralized systems, the data processing and decision making are done locally by each agent. Communication in the decentralized systems usually occurs before the decision making process in order to collect information. There is no communication between the agents while they are working on solving a problem locally or making decisions.

2.2.9 Fully Decentralized Systems

Fully decentralized systems are decentralized systems in which there is no leader or central authority at all either for organizing the data sharing between the agents or organizing the data processing locally by each agent. For example, if there is a leader that helps in initializing and sharing the data between the agents and each agent processes the data locally and takes a decision locally, then this is an example of a partially decentralized system or just a decentralized system but not a fully decentralized system.

2.2.10 The JADE Platform

General architecture of multi-agent system simulated using JADE software is shown in figure 2.4. As shown in this figure, the JADE platform consists of one or more containers, and each container contains some agents. Containers of a certain platform can be distributed over different computers as in platform 1 in figure 2.4 or they can exist on one computer. Each platform should have a special container called the Main Container. This container is the first container that starts in the platform, and then the other containers in the platform register with it. Each main container contains two special agents, the Directory Facilitator (DF) agent and the Agent Management System (AMS) agent. The DF agent is responsible for providing the yellow pages service. The AMS agent is responsible for managing the platform. It performs management actions like starting an agent, deleting an agent or shutting down the platform. All the agents in the platform register with the AMS agent that provides a directory for all the present agents in the platform and their current states. Another special agent that can be seen in a JADE platform is the Remote Monitoring Agent (RMA). This agent is responsible for implementing a graphical platform management console which provides a visual interface for monitoring the JADE platform as shown in figure 2.5.

2.3 Graph Theory Definitions and Concepts

To discuss the consensus algorithm, some graph theory definitions and concepts have to be discussed first.

A graph G (V, E) consists of a set of vertices V and a set of edges E that connects these vertices. The set of vertices can be defined as V= {v_1, v_2, ….., v_n} where n is the total number of vertices in the graph. The set of edges E can be defined as

$$E = \{(v_i,\ v_j)\ :\ v_i \in V\ \&\ v_j \in V\} \qquad (2.9)$$

A graph G (V, E) is directed if the set of edges consists of ordered set of vertices, and a graph is undirected if the set of edges consists of unordered set of vertices. A graph that contains no loops or multiple edges is called a simple graph. An undirected graph G is said to be connected if it has at least one path between any two vertices in the graph. A directed graph or digraph is said to be strongly connected if it has at least one directed path between any two vertices in the graph. Any connected graph without loops is called a tree, and the tree that contains all the vertices of the graph is called a spinning tree. The neighbors of a vertex v_i in the graph G (V, E) are defined as

$$N_i = \{v_j : (v_i, v_j) \in E\} \tag{2.10}$$

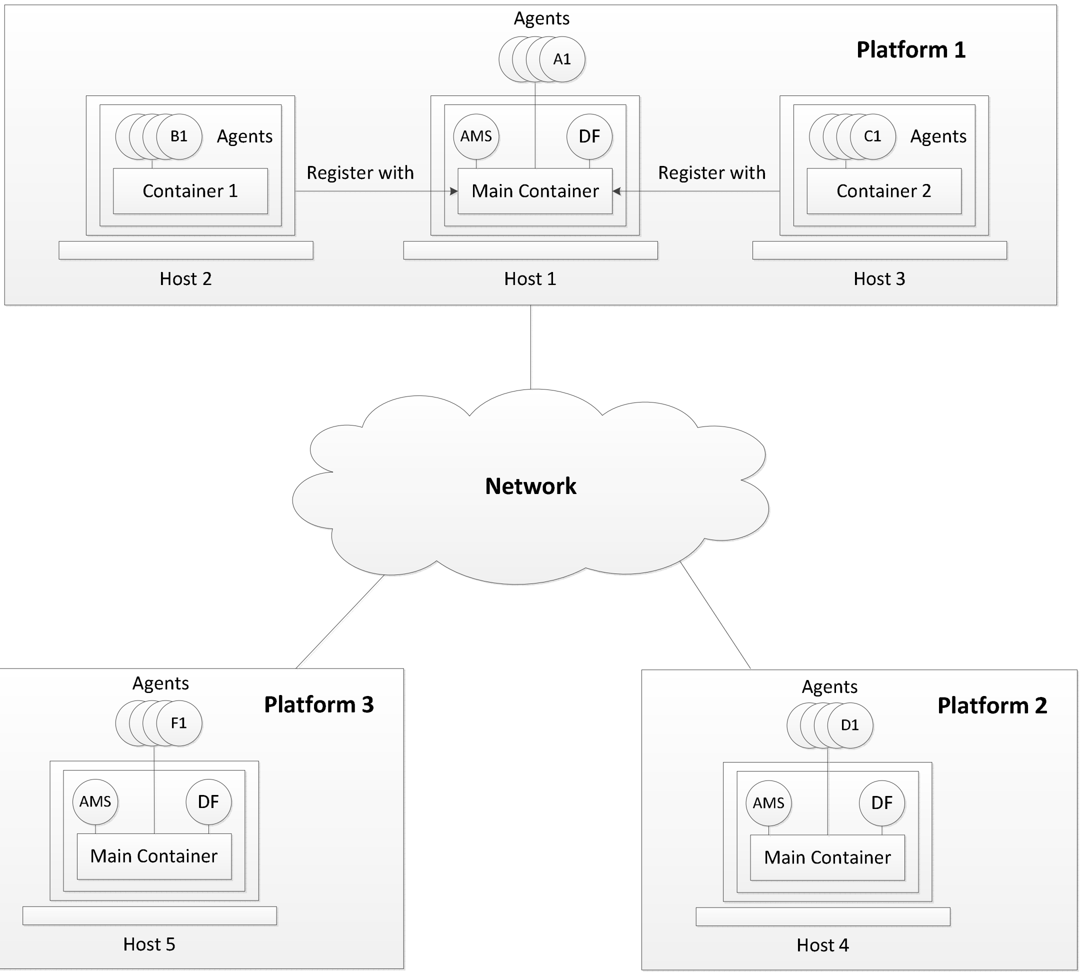
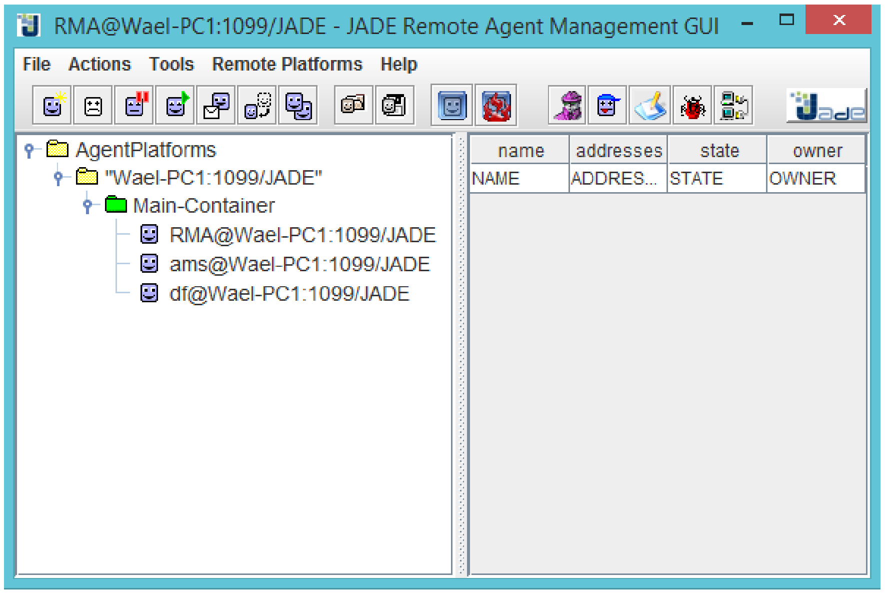


Figure 2.4 General architecture of multi-agent system simulated using JADE

The degree of a vertex in a graph is the number of edges that have this vertex in common. It is donated by deg (v_i). The degree matrix of a graph is $n \times n$ diagonal matrix D with the diagonal elements D_{ii} = deg (v_i).

$$D = \begin{pmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_{nn} \end{pmatrix} \tag{2.11}$$



AgentPlatforms
"Wael-PC1:1099/JADE"
Main-Container
RMA@Wael-PC1:1099/JADE
ams@Wael-PC1:1099/JADE
df@Wael-PC1:1099/JADE

name	addresses	state	owner
NAME	ADDRES...	STATE	OWNER

Figure 2.5 GUI interface for monitoring the JADE platform

The incidence matrix A in the case of undirected graph with m vertices and n edges is an $m \times n$ matrix in which a_{ij} = 1 if edge e_j is incident on vertex v_i and zero otherwise. The incidence matrix A in case of directed graph with m vertices and n edges is an $m \times n$ matrix in which a_{ij} = 1 if edge e_j enters vertex v_i, a_{ij} = -1 if edge e_j leaves vertex v_i and a_{ij} = 0 otherwise. Adjacency matrix C of graph G with n vertices is $n \times n$ matrix with elements c_{ij}=1 if vertex v_i is adjacent to vertex v_j and zero otherwise. The difference between the degree matrix D and the adjacency matrix A is defined as the laplacian matrix L. The second smallest eigenvalue of L is the algebraic connectivity of the graph.

$$L = D - A \tag{2.12}$$

2.4 The Consensus Algorithm

The consensus algorithm can be defined as an algorithm that will lead to an agreement between the agents about a certain state value. It is said that the agents of a network have reached a consensus if and only if $x_i = x_j$ for all i, j.

2.4.1 The Average Consensus Algorithm

The best known consensus algorithm is the average consensus algorithm. The continuous-time version of the average consensus algorithm can be written as

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}(x_j(t) - x_i(t)) \tag{2.13}$$

Where a_{ij} is the (i,j) element of the adjacency matrix. The above dynamics can be written in more compact form as follows:

$$\dot{x} = -Lx \tag{2.14}$$

Where L is the graph Laplacian matrix. The solution of the above set of differential equations is

$$x_i = \frac{1}{n}\sum_{i} x_i(0) \qquad i = 1, 2, \cdots, n \tag{2.15}$$

The discrete-time version of the average consensus algorithm is given by [9]

$$x_i(k+1) = x_i(k) + \omega \sum_{j \in N_i} a_{ij}(x_j(k) - x_i(k)) \tag{2.16}$$

The above equation can be written in the following compact form

$$x(k+1) = P\,x(k) \tag{2.17}$$

Where ω is the step size, and P is the Perron matrix and equal $I - \omega L$, where I is the identity matrix and L is the laplacian matrix.

2.4.2 The Minimum Consensus Algorithm

In the minimum consensus algorithm, the final state value of all the agents would be equal to the smallest initial value among them. The minimum consensus algorithm is achieved through the following protocol [10]

$$x_i(k+1) = \min\{x_i(k), \min_{j \in N_i} x_j(k)\} \tag{2.18}$$

2.4.3 The Maximum Consensus Algorithm

In the maximum consensus algorithm, the final state value of all the agents would be equal to the largest initial value among them. The maximum consensus algorithm is achieved through the following protocol [10]

$$x_i(k+1) = \max\{x_i(k), \max_{j \in N_i} x_j(k)\} \tag{2.19}$$

2.4.4 The Consensus on the most up-to-date Information

In the consensus on the most up-to-date information, the algorithm is based on associating timestamp information to the state value of each agent [4]. This enables each agent to store the most up-to-date information it receives from the other agents. Each agent carries a state vector x_i and a timestamp vector t_i where

$$\begin{aligned} t_i &\in R^n \\ x_i &\in R^n \end{aligned} \tag{2.20}$$

Where n is the total number of agents.
x_{ii} is the element in the vector x_i that carries the state of bus i evaluated at bus i.
x_{ij} is the element in the vector x_i that carries the state of bus j evaluated at bus i.

The values of x_{ij} are updated with the most up-to-date information using the timestamp t_{ij} associated to x_{ij}. The protocol used by each agent to achieve consensus on the most up-to-date information is presented in [4].

2.5 Recent Advances in Developing Decentralized and Distributed Solutions for the EDP

2.5.1 Solving the EDP using a Consensus on Lambda Approach

In [2], the authors proposed a novel consensus based distributed algorithm to solve the EDP. The algorithm can be implemented in a fully distributed system. The algorithm assumes that a strongly connected communication graph connects the agents for data exchange. The algorithm enables the agents to collectively learn the mismatch between the total power generation and total demand. This mismatch is used as a feedback to adjust the current power generation at each node. The algorithm succeeded to minimize the total cost while satisfying the power balance constraints. This proposed algorithm has the following disadvantages: it neglects the transmission losses, and it neglects all the practical non-convex cost functions that may be encountered in real systems. The authors assumed a directed graph to represent the information flow between the agents and they considered this as an advantage since it is a less restrictive assumption compared to undirected graphs; However, from the reliability point of view, this is not an advantage; a failure in any of the directed links that breaks the path of updating the information of a certain agent will lead to a failure for the algorithm in cases where this path is the only path to update the state of that agent.

In [1], a decentralized solution using self-organizing dynamic agents is proposed for solving the economic dispatch problem. In this paper, a network of cooperative dynamic agents has been used to compute the global quantities needed to solve the economic dispatch problem through utilizing the average consensus algorithm. In [1], the agents utilized the concept of the mutually coupled oscillators to reach a consensus. The mutually coupled oscillators or the mutually coupled dynamical systems can be implemented in a system comprised of two agents using two connected first-order dynamical systems, one at each node. A possible implementation of the mutually coupled dynamical systems is discussed in section 4.4.2 of this thesis. The disadvantages of the proposed approach in [1] are that it cannot be extended to solve the NCEDP; in addition, the approach is not fully decentralized: it requires assistance of the central authority for incorporating the transmission losses and for calculating the total number of agents.

In [3], a leader-follower consensus algorithm is proposed. The algorithm requires a leader agent. This agent is responsible for updating the incremental cost variable based on the power mismatch, and then the follower agents follow this incremental cost variable using the basic discrete time consensus algorithm. The algorithm has applied successfully for solving the EDP with convex cost functions. The power losses in the lines are not considered. The authors assumed that the power mismatch is already known by the leader agent. The algorithm has been investigated for different network topologies. The convergence rates of both the total power generation and the incremental cost variables have been studied under different network topologies and different number of generation units. In addition to the disadvantages of having a leader agent and neglecting the transmission losses, the proposed approach in [3] assumes only economic dispatch problem with convex cost functions.

In [4], a distributed approach is proposed for solving the economic dispatch problem in which the transmission losses and the generator constraints are considered. This proposed approach consists of two consensus algorithms running in parallel. The first algorithm is the lambda consensus algorithm in which all the agents reach a consensus on the incremental cost variable. The incremental cost of each generation unit is updated based on the incremental cost of the neighbor units and a correction term based on the total power mismatch. The second algorithm is called the consensus on the most up-to-date information. This algorithm enables the agents to reach a consensus on the power mismatch. The transmission losses have been incorporated in computing the total power mismatch. The authors treated the losses as constant from one dispatching period to the next while solving the economic dispatch problem. As the units are dispatched based on a new system load for the next dispatching period rather than on the current system load, the power flow in the network will change and the total losses will change; however, the authors assumed that this change between two consecutive dispatching periods can be neglected. The proposed approach in [4] works only for solving the economic dispatch problem with convex cost functions.

In [11]-[14], the consensus on the incremental cost variable has been applied for solving the EDP with convex cost functions. In these references, a leader agent is required. The disadvantage of the existence of a leader agent is that it preserves the single point failure issue that exists in centralized systems.

To summarize, all the previously proposed decentralized and distributed algorithms that utilized the consensus on lambda approach for solving the EDP cannot be extended for solving the NCEDP and lack the suitable mechanism for incorporating the transmission losses in an appropriate or accurate way. In addition, some of these algorithms require a leader agent for their operation.

2.5.2 Solving the EDP using a Constrained Distributed Gradient Algorithm

An improved distributed gradient algorithm is proposed for solving the EDP in [6]. This algorithm has been applied for solving the EDP problem with both equality and inequality constraints. The inequality constraints are the lower and upper bounds of the generators. The equality constraints are addressed based on a properly designed updating rule, and the inequality constraints are handled through reconfiguring the virtual communication topology. The N-1 rule has been considered during designing the communication network, so the algorithm can withstand the failure of one communication link. The main disadvantages of this algorithm are the same as those of the consensus on lambda approach. This algorithm can only be used for solving the EDP problem with smooth cost functions. The transmission losses also are not considered in [6].

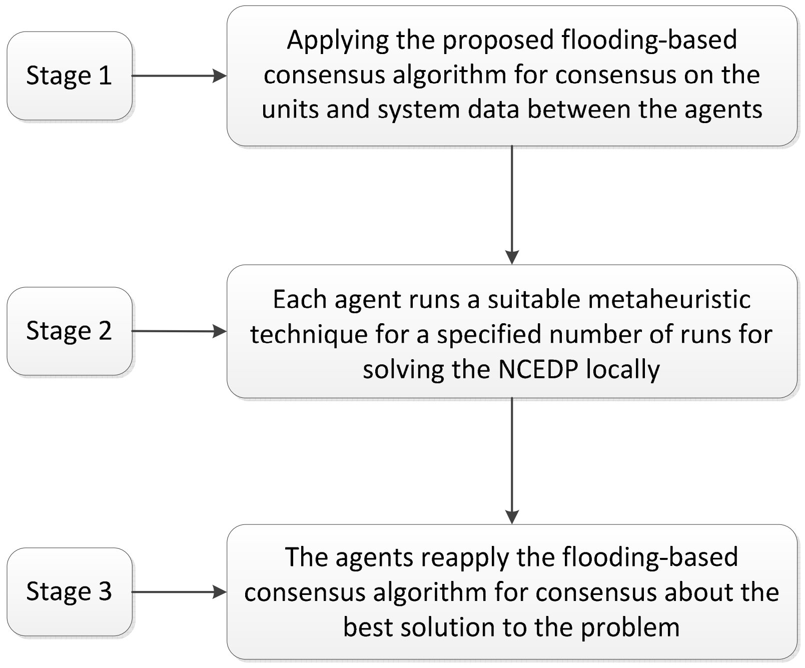
2.5.3 Solving the EDP using an Auction-based Distributed Algorithm

The authors in [5] presented a distributed algorithm that utilizes an auction technique called exchange. The proposed algorithm has been used in solving the non-convex economic dispatch problem. The valve-point loading, the multiple fuel options, the prohibited operating zones have been taken into consideration. Since the economic dispatch problem with the transmission losses is analytically intractable [5], the authors used an approximation to the transmission losses using Kron's formula. The algorithm consists of two levels. In the first level, each agent computes two bids; one is the prices of increasing its generation power with specified predefined values, and the second one is the prices of decreasing its generation power with specified predefined values. After all the agents compute their bids, the maximum consensus algorithm is applied so that the agents agree about the maximum bids among them. Once the agents reach a consensus on these maximum bids, the agents that proposed these bids are considered as winners. Then, these winners update their output power to decrease the total generation cost of the system. This procedure iterates until the optimal solution is reached and the economic dispatch problem is solved. The information flow between the agents is modeled by both a line graph and a complete graph. The agents communicate only with their neighbors in the graph and all the units reach a consensus on the highest bid after a number of iterations equal to the graph diameter.

The proposed approach in [5] assumes that each agent knows the total load of the system and the total number of generation units. This information is global and is not available for each agent. In addition, this algorithm is highly dependent on the initialization process. The algorithm has also the disadvantage of assuming constant loss coefficients or central authority assistance in computing the loss coefficients. Another inadequacy is that better-quality solutions for the NCEDP can be obtained if an efficient metaheuristic technique was used rather than the deterministic algorithm proposed in [5].

Chapter 3
The Proposed Approach

The approach presented in this thesis is proposed primarily for solving the NCEDP in a fully decentralized manner. In this case, a metaheuristic technique is used to solve the NCEDP locally by each agent, and the approach consists of three stages as shown in figure 3.1. With some adaptation, the proposed approach can also be applied for solving the EDP with convex cost functions in a fully decentralized manner as discussed in case study 4 in Chapter 4.



Stage 1
Applying the proposed flooding-based consensus algorithm for consensus on the units and system data between the agents
Stage 2
Each agent runs a suitable metaheuristic technique for a specified number of runs for solving the NCEDP locally
Stage 3
The agents reapply the flooding-based consensus algorithm for consensus about the best solution to the problem

Figure 3.1 The proposed approach for solving the NCEDP.

The agents will start the communication for collecting new data about the power network and solving the economic dispatch problem in a periodic time frame as shown in figure 3.2, where period 1 = period2 = period3 = period n.

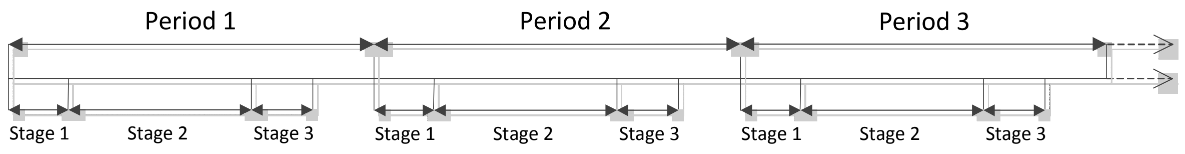


Figure 3.2 General time frame for solving the NCEDP.

The development of the fully decentralized approach proposed in this thesis entails the following assumptions:

- Each power generation station in the power system has an agent responsible for solving the EDP and this agent is embedded in the control system of the power station.
- Each bus considered by the proposed approach must have a distinct number that characterizes it from other buses considered.
- The system load can be aggregated for a specific set of buses in which each bus is responsible for estimating or forecasting its connected load for the next dispatching period. This assumption has been implicitly taken into account in previous reported studies in which fully distributed algorithms are proposed, such as [2], [4], and [5]. For future power systems, i.e., smart grids, a set of intelligent buses for which the total system load can be aggregated, with the buses estimating and forecasting their load and then communicating with one another, is a reasonable assumption.
- An undirected ring communication graph is assumed to connect the agents.

Assuming a ring topology is not a restrictive assumption. The advantage of the ring graph is that it provides a simple stopping criterion for the proposed flooding algorithm, as explained later in this Chapter. For connecting the agents, any other undirected and connected communication graph can be assumed. A general stopping criterion that can be used with any graph topology is a specific predefined time period during which the agent will wait without receiving any new message. Each agent communicates only with its neighbors in the graph.

3.1 Stage 1: Sharing the Required Data between the Agents using the Flooding-based Consensus (FBC) Algorithm

In the first stage, the agents reach consensus on the problem data, using a flooding-based consensus (FBC) algorithm. The FBC algorithm is based on the concept of flooding algorithms [15], [16], which are routing algorithms used in data networks for broadcasting messages or data packets between network nodes. Flooding algorithms, whose application in computer networks and ad hoc wireless networks is known, function in the following way. The source node sends its data to its neighbors; each node in the network that receives the data then stores a copy of these data and resends them to all of its neighbors except the one from which it received the data. Each node sends the data it has received only once. If a specific network node receives repeated data, it ignores these data. The algorithm stops when all nodes have received and transmitted the data only once. Since a ring communication graph is assumed in this work, each agent will receive a repeated message only if the message has already been transmitted by all of the other nodes in the network. This condition can be used as a stopping criterion for each agent.

Each message contains a bus stamp element that is used as follows. First, when each agent prepares its message, it adds a bus stamp element equal to its bus number. Then, Any agent in the network that receives a message changes the bus stamp element in the message to match its own bus number before relaying the message to its neighbors. The bus stamp enables each agent to determine which neighbor sent the message and to which neighbor it must transmit this message.

When each agent receives a message that contains unit data, it copies and stores it in a dynamic matrix. Figure 3.3 shows a flowchart that clarifies the operation of the FBC algorithm; D_i is the message that contains the data from agent i, and N_i is the set of neighboring agents for agent i. An example of this message is a tuple $D_i = \langle bus_{nu.}, bus_{st.}, x_i \rangle$, where $bus_{nu.}$ is the bus number; $Bus_{st.}$ is the bus stamp; and x_i is a vector that contains the units data such as the cost coefficients and the bus load.

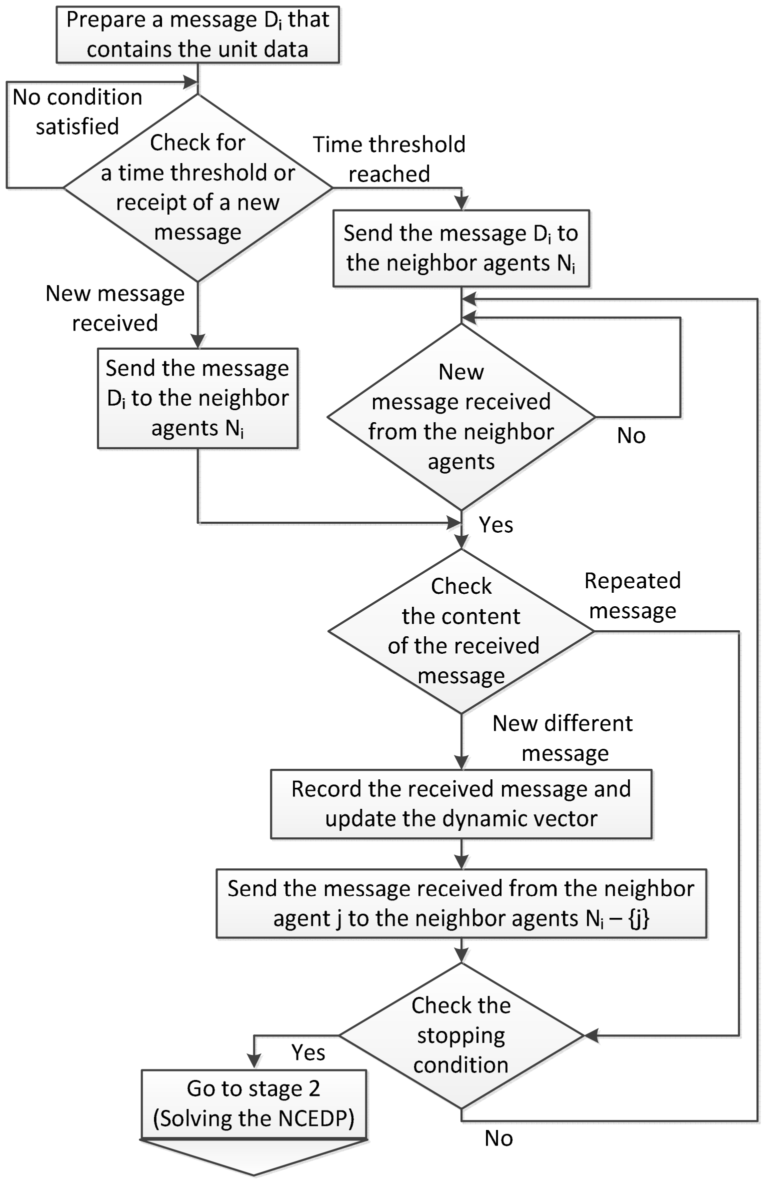
3.2 Stage 2: Solving the NCEDP in Parallel using a Suitable Metaheuristic Technique

After the agents have reached a consensus with respect to the problem data, the second stage begins, during which each agent solves the EDP locally using a suitable algorithm. For solving the NCEDP, an appropriate metaheuristic technique can be applied during this stage. A general flowchart that clarifies the operation of the second stage that involves a metaheuristic technique is shown in Figure 3.4. In this figure, nr is the run counter, nrt is the total number of runs, and g is the iteration counter.

The differential evolution algorithm has been chosen for solving the NCEDP by each agent. The differential evolution algorithm is one of the most efficient metaheuristic techniques that have been proposed for solving the NCEDP. The results obtained by the differential evolution algorithm with the constraints handling techniques adapted in this thesis were found to be satisfactory compared to those presented in the previous literature. The differential evolution is a stochastic search method that works in the general framework of the evolutionary algorithms. The operators used by the differential evolution are the mutation, crossover and selection. The differential evolution algorithm starts by initializing the population vectors within their limits. For each vector in the population X_i, three other vectors X_j, X_k and X_m are selected randomly from the population such that $j \neq k \neq m$. Thereafter, these three vectors are employed to create a new mutated vector U_i as follows:

$$U_i = X_j + F \cdot (X_k - X_m) \tag{3.1}$$

Where F is a scaling factor.



Prepare a message D_i that contains the unit data
No condition satisfied
Check for a time threshold or receipt of a new message
Time threshold reached
Send the message D_i to the neighbor agents N_i
New message received
Send the message D_i to the neighbor agents N_i
New message received from the neighbor agents
No
Yes
Check the content of the received message
Repeated message
New different message
Record the received message and update the dynamic vector
Send the message received from the neighbor agent j to the neighbor agents N_i – {j}
Check the stopping condition
Yes
No
Go to stage 2 (Solving the NCEDP)

Figure 3.3 Stage 1: Flooding-based Consensus (FBC) Algorithm.

After the previous mutation process, the binomial crossover process is applied for mating the vector U_i and the vector X_i according to the crossover probability C to produce a new offspring Z_i. The crossover operation can be expressed as

$$Z_{ij} = \begin{cases} U_{ij} & if\ rand(j) \leq C \\ X_{ij} & otherwise \end{cases} \tag{3.2}$$

Where Z_{ij}, U_{ij} and X_{ij} are the j-th components of the vectors Z_i , U_i and X_i respectively, rand (j) is the j-th evaluation of a uniformly distributed random variable between 0 and 1.

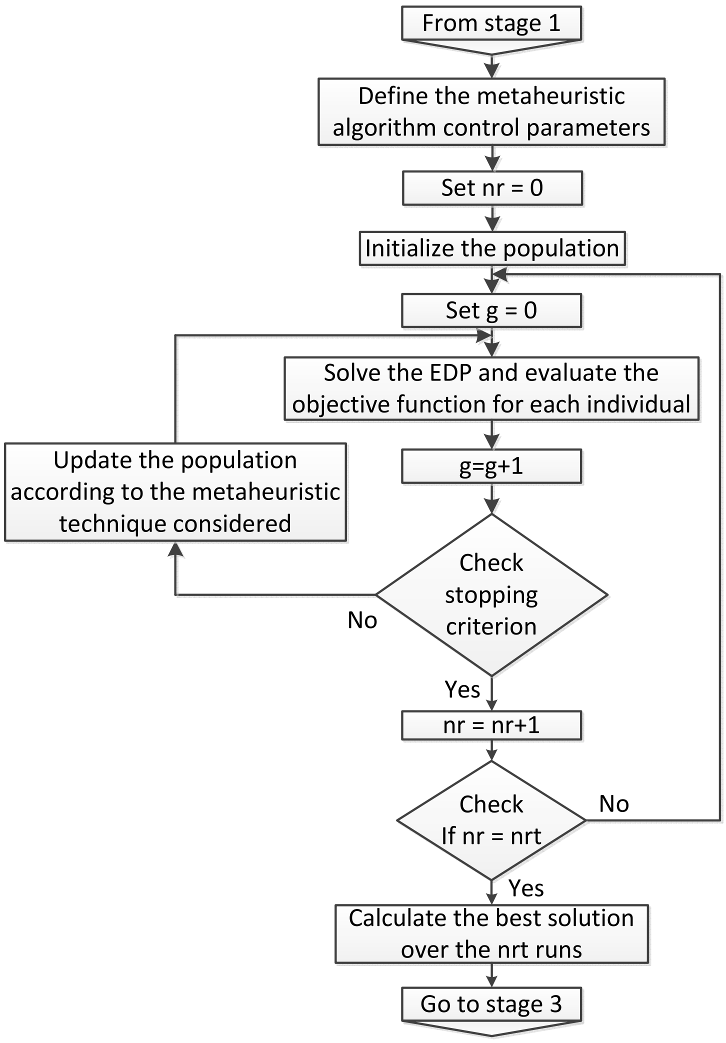

From stage 1
Define the metaheuristic algorithm control parameters
Set nr = 0
Initialize the population
Set g = 0
Solve the EDP and evaluate the objective function for each individual
g=g+1
Update the population according to the metaheuristic technique considered
Check stopping criterion
No
Yes
nr = nr+1
Check If nr = nrt
No
Yes
Calculate the best solution over the nrt runs
Go to stage 3

Figure 3.4 Stage 2: General flowchart of metaheuristic technique for solving the NCEDP locally.

Finally, the selection process is applied to choose one survivor between the vector X_i and the vector Z_i for the next generation. If the fitness value corresponding to Z_i is higher than that of X_i, then Z_i will replace X_i in the next generation, otherwise X_i will remain the same for the next generation.

In order to initialize the population, each component of the individuals is generated randomly as follow

$$P_i = P_i^{\min} + (P_i^{\max} - P_i^{\min}) \times \mathrm{rand} \tag{3.3}$$

Where rand is a random number between 0 and 1 that obeys the uniform distribution. For handling inequality constraints like the prohibited operating zone, the generator limits and the ramp rate limits, the penalty function method has been used. In this method, if the generators output power has violated any of the above inequality constraints, the problem solution that contains this output power is penalized with a very large positive constant. For handling the equality constraints, the concept of the dependent source in [17] and [18] has been used in which a dependent unit is chosen randomly, and then the power mismatch is added to its output power. If this dependent unit violates its generation limits due to the mismatch added to it, then the unit is fixed to its violated limit, and the remaining of the mismatch is compensated by another randomly selected unit. This process continues until the mismatch vanishes. In the case of a system in which the transmission losses are not considered, the mismatch is computed as follows:

$$mismatch = P_D - \sum_{i=1}^{n} P_i \tag{3.4}$$

In the case of considering the transmission losses, the mismatch is computed as follows:

$$mismatch = P_D + P_L - \sum_{i=1}^{n} P_i \tag{3.5}$$

3.3 Stage 3: Consensus on the Final Solution to the Problem

Since metaheuristic techniques are stochastic in nature, with each run of a metaheuristic algorithm possibly producing a different solution, the agents will produce different solutions to the problem. To address this issue, a third stage is incorporated to enable the agents to reach a consensus on the best solution to the problem. Figure 3.5 shows a flowchart that illustrates stage 3, during which each agent prepares a message S_i that contains the best solution obtained by that agent. An example of this message is a tuple $S_i = \langle bus_{nu.}, bus_{st.}, x_i \rangle$, where bus$_{nu.}$ is the bus number; Bus$_{st.}$ is the bus stamp; and x$_i$ is a vector defined as $x_i = [Total\ \cos t_i, P_{1,\ldots,} P_n]$, where total cost$_i$ is the best cost computed by agent i, and P$_1$,..., P$_n$ are the corresponding power outputs produced by the generation units. After the agents apply the FBC algorithm so that they can reach consensus on the best solutions, each agent computes a final solution to the problem using the following equations:

$$i^* = \arg\min_{i \in n} (Total \cos t_i) \quad (3.6)$$

$$x_i = x_{i^*} \quad (3.7)$$

where n is the total number of generation units in the network, and i* is the generator index that provides the best solution to the problem.

3.4 Incorporating the Transmission Losses

To compute transmission losses in a fully decentralized manner, the loss coefficients in (2.4) must be computed locally by each agent, which means that each agent must solve the power flow problem locally [19]. This step can be performed, if each agent has knowledge of the complete power network data represented in the transmission line data and bus data, in the following way. It is assumed that each agent knows its own bus data plus the data related to the transmission lines connected to its bus, and each agent then prepares a message that contains a data matrix in which the first row contains the bus number, the bus stamp, and the generator data. The second row contains bus data such as the bus type; whether it is a slack, constant power, or load bus; the bus voltage; and the active and reactive load power. The next rows, one row per transmission line, contain the data of the transmission lines connected to the bus: the resistance, the reactance, and one-half of the total line susceptance. After this message is prepared, the FBC algorithm is applied, stopping when each agent has all the data it needs to solve the power flow problem locally. At this stage, each agent can include power flow equations as constraints while solving the EDP; however, handling power flow equations in a metaheuristic technique demands substantial computational power. For this reason, after the power flow problem has been solved locally, the loss coefficients are computed from the power flow results as discussed in [19]. Kron’s loss formula is then applied by each agent in order to calculate the transmission losses for each candidate solution in the metaheuristic algorithm.

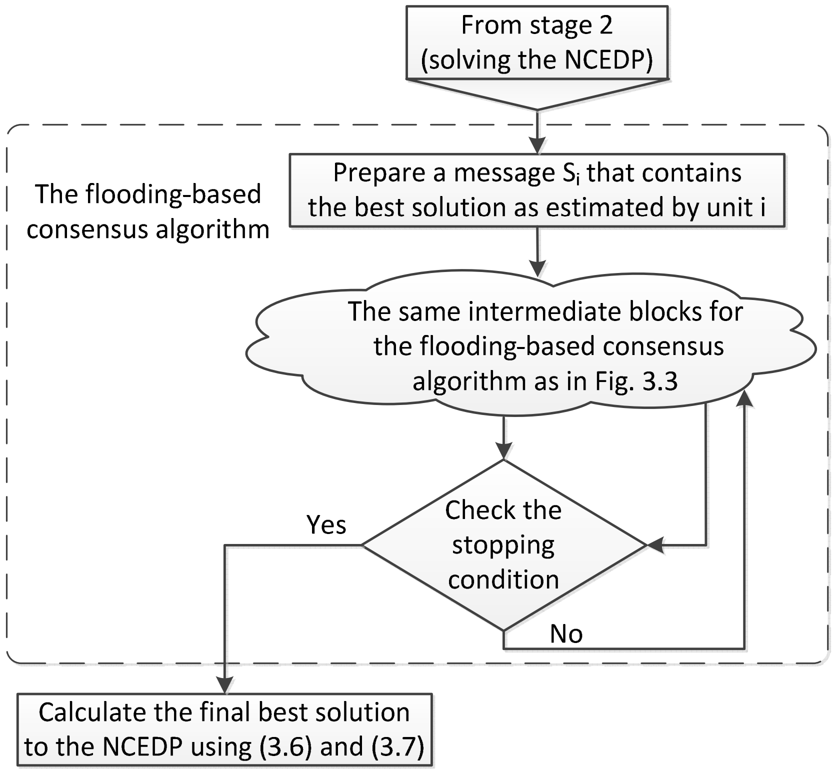

From stage 2 (solving the NCEDP)
The flooding-based consensus algorithm
Prepare a message S_i that contains the best solution as estimated by unit i
The same intermediate blocks for the flooding-based consensus algorithm as in Fig. 3.3
Check the stopping condition
Yes
No
Calculate the final best solution to the NCEDP using (3.6) and (3.7)

Figure 3.5 Stage 3: Consensus on the best solution to the NCEDP.

Chapter 4
Case Studies and Simulation Result

The first case study in this Chapter provides some experimentation with JADE software for the purpose of investigating the operation and simulation of multi-agent systems, and for the purpose of approximating the required communication time by the proposed approach. The second and third case studies provide examples of the application of multi-agent systems and data networks through the proposed approach for solving the NCEDP. The last case study which consists of four parts investigates four of the previously proposed approaches for solving the EDP and compares them with the proposed approach in this thesis.

4.1 Case Study 1: Experimentation with JADE Software and Approximating the Communication Time

4.1.1 Experimentation with JADE Software

In order to approximate the required communication time by the proposed approach, JADE software has been used. This section provides some experimentation related to the simulation of multi-agent systems using JADE, and the following section explains the methodology used to approximate the required communication time by the proposed approach.

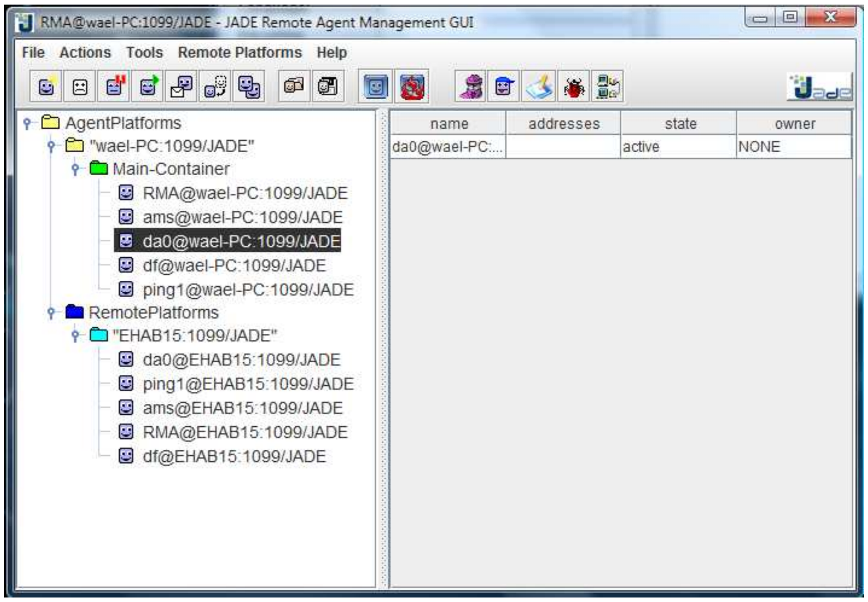
In order to simulate the multi-agent system over different platforms, communication medium is required to carry the messages and data between agents. In this study, the internet is used to enable communication between agents on different and distance-apart platforms. The advantage of using the internet is that it does not force any constraints on the distance between the platforms that carry the agents. In most of the previous literature that study the employment of multi-agent technology to solve smart grid challenges, the whole multi-agent system was simulated on a single personal computer and the communication time delay is not provided or neglected. This does not clarify or highlight the situation of some of the practical problems such as, for example, will the existence of communication between the agents that are distributed over different and distance-apart platforms introduce undesirable or extra time delay in the system operation or what is the probability that an error, deterioration to the message content, will happen during transferring a message between two agents through certain communication medium.

To provide some examples of real distributed multi-agent systems, three personal computers have been used. The first computer has a name “EHAB15” and is located inside

University of Waterloo, the other two computers namely; “wael-PC” and “Wael-PC1” are located at a house approximately 2.4 km from EHAB15 computer. JADE software has been used for modeling the multi-agent system. The internet has been used to establish the connection between the agents on wael-PC and those on EHAB15. Two of the JADE demo agents have been launched on each of the PCs. The first agent is a dummy agent with local name “da0”. This agent provides the capability of displaying the messages sent and received by it using a GUI. The other agent is called the ping agent. This agent is one of the demo agents available in JADE software. The function of this agent is to respond to the message that contains the word “ping” and has a communicative act “query-ref” with a message contains the word “alive” and has a communicative act “inform”. These two agents, the dummy agent and the ping agent, are the simplest agents that can be used to test the communication between two platforms. Figure 4.1 shows a screenshot of the JADE Remote Agent Management GUI from wael-PC, and Figure 4.2 shows a screenshot of the JADE Remote Agent Management GUI from EHAB15. It is clear from figures 4.1 and 4.2 that each platform is able to detect the existence of the agents on the other remote platform through the internet.

To check the probability of error occurrence during sending a message between two agents, a message has been sent from the agent da0@EHAB15 to the agent da0@wael-PC. This message contains a 100 row by 100 column matrix that contains 10,000 numbers generated randomly with Matlab. Each number is represented by 16 digits which gives a total of 160,000 digits. The received matrix on wael-PC has been compared with the original sent matrix, and it has been found that the 160,000 digits are exactly the same. Yet, another message has been sent that contains 250 x 250 matrix in which each element is represented by 16 digits. This gives a total of 1 million digits in one single message. By comparing the sent and received matrices, the matrices were found to be exactly the same without any difference in the 1 million digits.

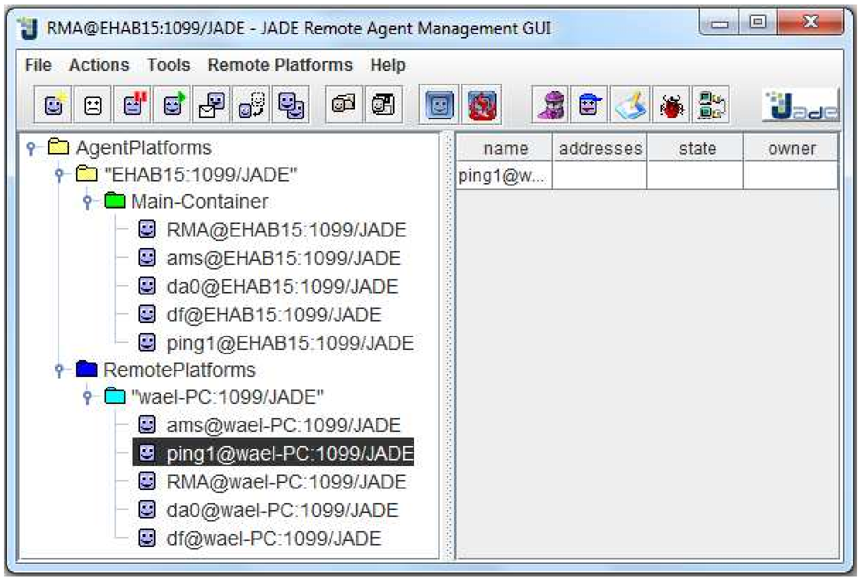
One of the JADE tool agents is an agent called the sniffer agent, this agent is usually used in the literature to display a record of the messages sent and received inside the same platform. To test this agent, several messages have been sent from the dummy agent at wael-PC to both the ping1 agent at the same platform and the Ping1 agent at EHAB15. Figure 4.3 shows a record of these messages by the dummy agent GUI and figure 4.4 shows the sniffer agent GUI which presents the corresponding record of these messages.



AgentPlatforms
"wael-PC:1099/JADE"
Main-Container
RMA@wael-PC:1099/JADE
ams@wael-PC:1099/JADE
da0@wael-PC:1099/JADE
df@wael-PC:1099/JADE
ping1@wael-PC:1099/JADE
RemotePlatforms
"EHAB15:1099/JADE"
da0@EHAB15:1099/JADE
ping1@EHAB15:1099/JADE
ams@EHAB15:1099/JADE
RMA@EHAB15:1099/JADE
df@EHAB15:1099/JADE

name	addresses	state	owner
da0@wael-PC:...		active	NONE

Figure 4.1 Screenshot for the agent platforms from wael-PC

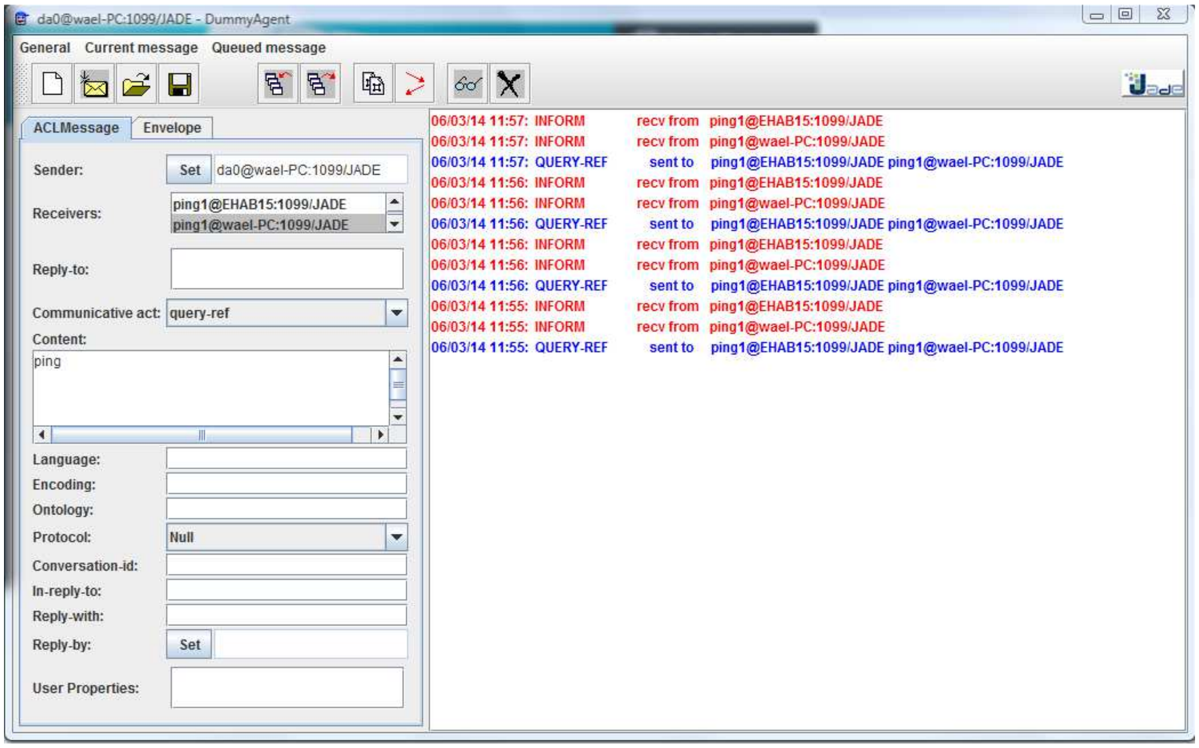


AgentPlatforms
"EHAB15:1099/JADE"
Main-Container
RMA@EHAB15:1099/JADE
ams@EHAB15:1099/JADE
da0@EHAB15:1099/JADE
df@EHAB15:1099/JADE
ping1@EHAB15:1099/JADE
RemotePlatforms
"wael-PC:1099/JADE"
ams@wael-PC:1099/JADE
ping1@wael-PC:1099/JADE
RMA@wael-PC:1099/JADE
da0@wael-PC:1099/JADE
df@wael-PC:1099/JADE

name	addresses	state	owner
ping1@w...			

Figure 4.2 Screenshot for the agent platforms from EHAB15 PC.

It is clear from comparing figures 4.3 and 4.4 that the sniffer agent is able to record the messages sent and received inside the same platform very well and is able to record the messages sent by the dummy agent to outside the platform; however, it failed to record the received messages from outside the platform.



Sender: Set da0@wael-PC:1099/JADE
Receivers: ping1@EHAB15:1099/JADE ping1@wael-PC:1099/JADE
Reply-to:
Communicative act: query-ref
Content: ping
Language:
Encoding:
Ontology:
Protocol: Null
Conversation-id:
In-reply-to:
Reply-with:
Reply-by: Set
User Properties:

```
06/03/14 11:57:  INFORM      recv from  ping1@EHAB15:1099/JADE
06/03/14 11:57:  INFORM      recv from  ping1@wael-PC:1099/JADE
06/03/14 11:57:  QUERY-REF     sent to  ping1@EHAB15:1099/JADE ping1@wael-PC:1099/JADE
06/03/14 11:56:  INFORM      recv from  ping1@EHAB15:1099/JADE
06/03/14 11:56:  INFORM      recv from  ping1@wael-PC:1099/JADE
06/03/14 11:56:  QUERY-REF     sent to  ping1@EHAB15:1099/JADE ping1@wael-PC:1099/JADE
06/03/14 11:56:  INFORM      recv from  ping1@EHAB15:1099/JADE
06/03/14 11:56:  INFORM      recv from  ping1@wael-PC:1099/JADE
06/03/14 11:56:  QUERY-REF     sent to  ping1@EHAB15:1099/JADE ping1@wael-PC:1099/JADE
06/03/14 11:55:  INFORM      recv from  ping1@EHAB15:1099/JADE
06/03/14 11:55:  INFORM      recv from  ping1@wael-PC:1099/JADE
06/03/14 11:55:  QUERY-REF     sent to  ping1@EHAB15:1099/JADE ping1@wael-PC:1099/JADE
```

Figure 4.3 Record of messages by the dummy agent GUI.

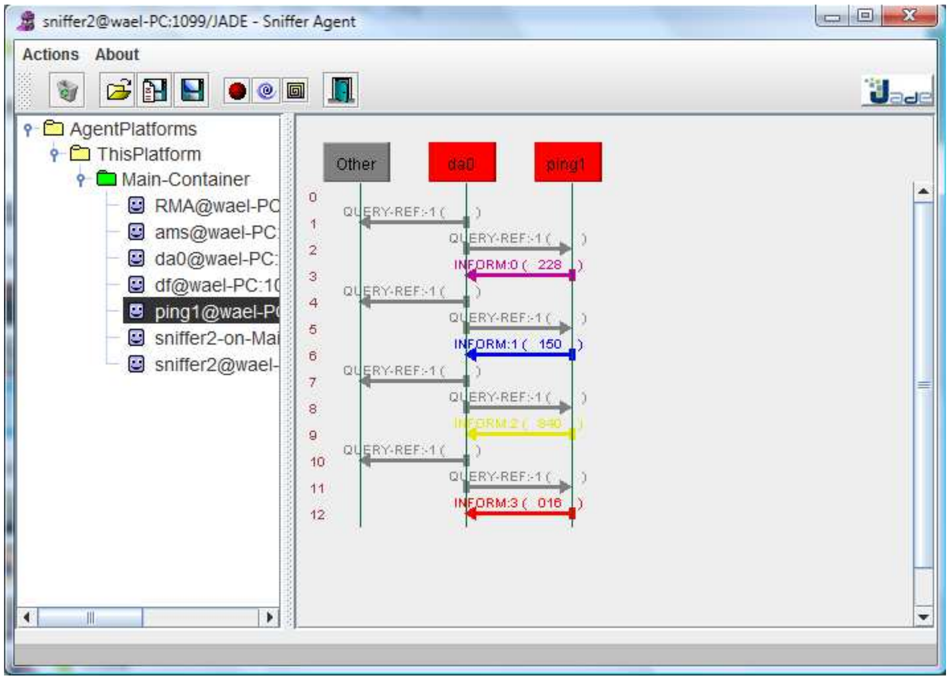
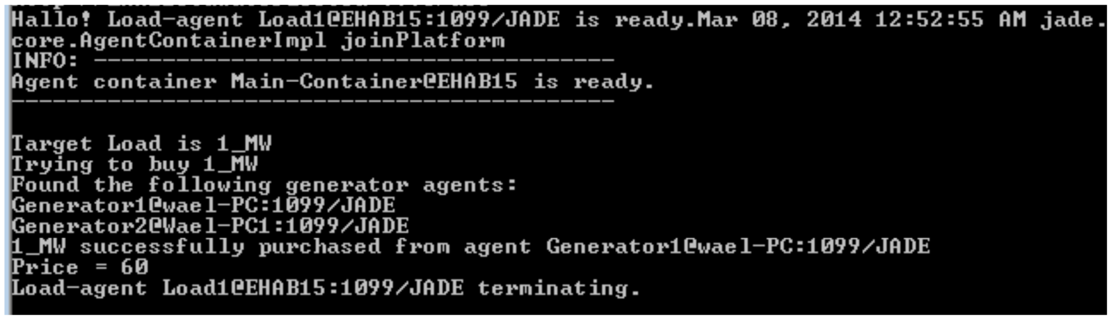


Figure 4.4 The sniffer agent GUI

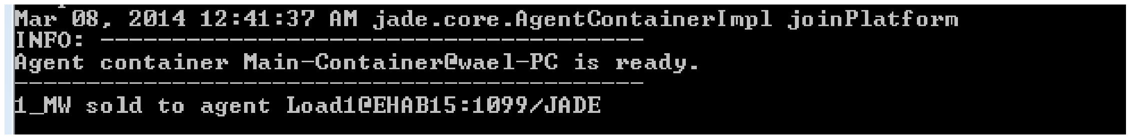
The last experiment in this section is an experiment with the DF agent. The function of the DF agent in multi-agent systems is to provide the yellow page services for the other agents. These yellow page services enable the agents to know about the other agents in the network that provide the service they need. For example, if a new generator agent wants to know the other generator agents that currently exist in the network and participate in solving the economic dispatch problem, it can know that through searching in the yellow pages about the agents which register their service as “dispatching the units”. It can also search the yellow pages about agents that register their service as “forecasting the load” in order to get their addresses from the yellow pages and communicate with them to calculate the total load of the system. In this study, the yellow page services have been used successfully between agents distributed over different platforms and communicate together using the internet. To demonstrate this, a simple example is presented. Although this example does not involve realistic numbers and may not reflect a corresponding real application in a direct way, it provides an example of the successful use of yellow page services over the internet. It is assumed that there are three agents, one agent is called load1 agent and is located at EHAB15. This agent wants to contract about purchasing 1 MW extra power. The other two agents; one is named Generator1 and is located at wael-PC and the second one is named Generator2 and located at Wael-PC1. The load1 agent does not know about the current existing agents in the network which provide the service of selling power, but it knows the addresses of the DF agents at wael-PC and at Wael-PC1. Therefore, it searches the yellow pages over the remote platforms and discovers that there are two agents called Generator1 and Generator2 which provide the service of selling power. Then, it gets these agents’ addresses from the yellow pages and start negotiating with them. The generator1 agent provides a price of 60 $ per 1 MW and generator2 agent provides a price of 70 $ per 1 MW. The load1 agent compares the prices and purchases the 1 MW from generator1. Figure 4.5 and 4.6 shows the summary of this process.



```
Hallo! Load-agent Load1@EHAB15:1099/JADE is ready.Mar 08, 2014 12:52:55 AM jade.
core.AgentContainerImpl joinPlatform
INFO: --------------------------------------
Agent container Main-Container@EHAB15 is ready.
--------------------------------------------

Target Load is 1_MW
Trying to buy 1_MW
Found the following generator agents:
Generator1@wael-PC:1099/JADE
Generator2@Wael-PC1:1099/JADE
1_MW successfully purchased from agent Generator1@wael-PC:1099/JADE
Price = 60
Load-agent Load1@EHAB15:1099/JADE terminating.
```

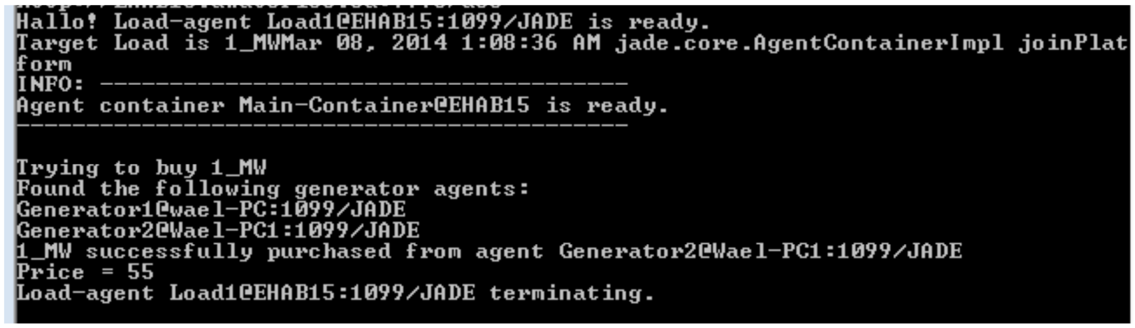
Figure 4.5 Screenshot from EHAB15 PC



```
Mar 08, 2014 12:41:37 AM jade.core.AgentContainerImpl joinPlatform
INFO: --------------------------------------
Agent container Main-Container@wael-PC is ready.
--------------------------------------------
1_MW sold to agent Load1@EHAB15:1099/JADE
```

Figure 4.6 Screenshot from wael-PC

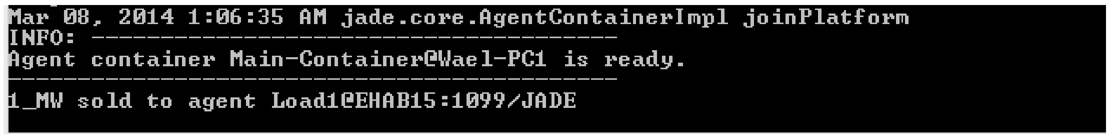
Now, the process has been repeated after reducing the price at Wael-PC1 (Generator2 agent) to 55 $ instead of 70$. Figure 4.7 and 4.8 shows the summary of this process.



```
Hallo! Load-agent Load1@EHAB15:1099/JADE is ready.
Target Load is 1_MWMar 08, 2014 1:08:36 AM jade.core.AgentContainerImpl joinPlat
form
INFO: --------------------------------------
Agent container Main-Container@EHAB15 is ready.
--------------------------------------------

Trying to buy 1_MW
Found the following generator agents:
Generator1@wael-PC:1099/JADE
Generator2@Wael-PC1:1099/JADE
1_MW successfully purchased from agent Generator2@Wael-PC1:1099/JADE
Price = 55
Load-agent Load1@EHAB15:1099/JADE terminating.
```

Figure 4.7 Screenshot from EHAB15 PC



```
Mar 08, 2014 1:06:35 AM jade.core.AgentContainerImpl joinPlatform
INFO: --------------------------------------
Agent container Main-Container@Wael-PC1 is ready.
--------------------------------------------
1_MW sold to agent Load1@EHAB15:1099/JADE
```

Figure 4.8 Screenshot from Wael-PC1

4.1.2 Approximating the Communication Time Required by the Proposed Approach

In order to approximate the communication time required by the proposed approach, the following experiment has been used. Two computers were located 2.4 km apart. An agent was launched on each computer using JADE software. The internet was used for the creation of the communication link between these two agents. To determine the time required to send a message between two agents, the time difference between the sending of the message by a specific agent and receipt of a response message by the same agent was recorded. This time is equivalent to the sending of two messages. With the above experimental setup, the average time required for a message to be sent between two agents was found to be 0.046 sec, which also includes the time required for JADE software to prepare for sending a message and to check the content of the received message. The approximation of the time needed for stage 1 and stage 3 is based on the assumption that stage 1 and stage 3 are completed through consecutive iterations. In each iteration, each agent receives two messages from its two neighbors, which also means that each agent sends two messages to those two neighbors per iteration. The total number of iterations required for the information to be shared among the agents can be computed as follows:

$$\text{Number of iterations} = \begin{cases} \dfrac{n}{2} & if\ \ n\ is\, even \\ \dfrac{n+1}{2} & if\ \ n\ is\, odd \end{cases} \tag{4.1}$$

Where n is the total number of nodes in the system. It is also assumed that the time required per iteration is equivalent to the time required for sending one message between two agents. The time required for sending a number of consecutive messages that matches the total number of iterations is assumed to be equal to the time required for stage 1. The same methodology was also applied for approximating the time required for stage 3. The total time required for solving the EDP using the proposed approach can be computed from the following formula:

$$total\ time = \sum_{i=1}^{4} t_i \tag{4.2}$$

Where t_1 is the time required for stage 1; t_2 is the time required for stage 2, as calculated using (4.3); t_3 is the time required for stage 3; and t_4 is the time required for solving the power flow problem and computing the loss coefficients, if needed.

$$t_2 = \max_{j}(time_j) \qquad j = 1, 2, \cdots, n \tag{4.3}$$

Where time_j is the total time required for agent j to solve the EDP for a specified number of runs, and n is the total number of generator agents. Table 4.1 shows the estimated time for stage 1 and stage 3 per each system used in the following case studies based on the measured communication time for sending one message and the total number of communication iterations.

4.2 Case Study 2: 26-Buses System with Transmission Losses, Ramp Rate Limits, and Prohibited Operating Zones

MATLAB has been used for validating the operation of the proposed approach. All of the metaheuristic technique runs were performed sequentially on one computer rather than in parallel on a number of computers. The operation of the FBC algorithm was evaluated based on data sharing between agents on the same computer; on the other hand, JADE software was used for approximating the total communication time delay as discussed in the previous case study. The system used in this case study consists of 26 buses, six thermal units, and 46 transmission lines. The ramp rate limits, the prohibited operating zones and the transmission losses are taken into consideration. The transmission line and bus data are as provided in [19], and the generator data are as given in [20].

TABLE 4.1
APPROXIMATED TIME FOR STAGE 1 AND STAGE 3

	Number of iterations	Time required by stage 1 (t_1) (sec.)	Time required by stage 3 (t_3) (sec.)
4-bus system	2	0.092	0.092
6-bus system	3	0.138	0.138
26-bus system	13	0.598	0.598
40-unit system	20	0.92	0.92

After the proposed approach was applied, all of the agents reached consensus on the solution shown in Table 4.2 under Differential Evolution (DE). To solve the power flow problem and to compute the loss coefficients, the MATLAB toolbox presented in [19] was used by each agent. A ring communication network connecting the 26 buses is assumed. It is also assumed that the generator buses are responsible for solving the EDP and each generator agent runs the differential evolution algorithm for three runs, so the total number of runs is thus 18. The control parameters used in the differential evolution algorithm are scaling factor = 0.09, crossover probability = 0.5, number of iterations = 200, and population size = 200. Table 4.2 shows the results obtained by the differential evolution algorithm and some of those reported in the literature. The total losses computed in [23]–[26] are lower than those obtained using Kron's loss formula. This discrepancy means that the total power generated as shown in those studies is lower than the actual value, which results in a lower total generation cost. When the prohibited operating zones and ramp rate limits are not considered, the resultant problem has convex cost functions and one optimal solution. This problem has been solved in [19], and the global optimal solution has a total cost equal to 15447.72 \$/hr. When the prohibited operating zones and ramp rate limits are considered, the global optimal solution should have a total generation cost either equal to 15447.72 \$/hr or higher than this value but not lower. From Table 4.2, it can be noted that the solution provided by the differential evolution algorithm has a mismatch equal to 0.02 MW, which is due to the rounding up of the power generated by each unit to two decimal places. This mismatch helps produce a slightly lower total cost, which might also be the case in [22], in which the power generation from each unit is also rounded up to two decimal places, and a mismatch of 0.01 MW exists. When the differential evolution algorithm is used without rounding up the output power computed for the units, the best solution found has a total cost of 15449.89 $/hr, with mismatch equal to -4.8e-10 MW. The loss coefficients used in all of the references cited in Table 4.2 are four decimal places rounded up from the original loss coefficients. When each agent solves the power flow problem and computes the original loss coefficients without rounding up, the solution obtained with the differential evolution algorithm entails a total cost of 15447.72369 $/hr, which is approximately equal to that provided in [19] when a convex cost function is considered. Hence, the ramp rate limits and the prohibited operating zones provided in [20] do not affect the optimal solution obtained when the cost function was convex. After each agent has computed the loss coefficients, the loss coefficients are rounded up to four decimal places in order to provide a fair comparison with the results from other studies. The total time required for the proposed approach to solve the above problem, as computed using (4.2) and (4.3), is 3.634 sec, where $t_1 = t_3 = 0.598$ sec, $t_2 = 2.064$ sec (maximum time for running the differential evolution algorithm for three runs over all of the generator agents), and $t_4 = 0.374$ sec.

4.3 Case Study 3: 40-Unit System with Valve-Point Effects

A 40-unit system was considered in this case study. The data of the system are listed in [27]. In this system, the cost function is non-convex due to valve-point effects. Each agent has used the differential evolution algorithm with the following control parameters; scaling factor = 0.09, crossover probability = 0.5, number of iterations = 1,500, and population size = 500. Each agent runs the differential evolution algorithm for 3 runs, so the total number of runs is 120. Table 4.3 shows a comparison between the results obtained using the differential evolution algorithm and those obtained by other algorithms in the literature for the 40 units system. All of the solutions shown in Table 4.3 are centralized except that from [5], which is distributed, and that obtained using the new approach which is decentralized. The algorithm proposed in [5] is the only attempt observed over the literature for solving the NCEDP in a distributed manner. The output power of each unit for the best solution obtained using the DE algorithm is provided in Appendix A.

TABLE 4.2

SIX-GENERATOR TEST SYSTEM: COMPARISON OF THE DE ALGORITHM SOLUTION WITH SOME OF THOSE REPORTED IN THE LITERATURE

Unit Power output	aBBOmDE [26]	RDPSO [25]	BBO [24]	SOH-PSO [23]	**DE**	GAAPI [22]	NPSO-LRS [21]	PSO [20]	GA [20]
P_1	447.3944	445.2541	447.3997	438.21	**448.27**	447.12	446.96	447.4970	474.8066
P_2	173.4968	172.7916	173.2392	172.58	**172.96**	173.41	173.3944	173.3221	178.6363
P_3	263.2259	263.5285	263.3163	257.42	**263.44**	264.11	262.3436	263.4745	262.2089
P_4	138.8915	141.0687	138.0006	141.09	**139.3**	138.31	139.512	139.0594	134.2826
P_5	165.1239	163.8578	165.4104	179.37	**165.28**	166.02	164.7089	165.4761	151.9039
P_6	87.2793	88.8558	87.07979	86.88	**86.68**	87.00	89.0162	87.1280	74.1812
Total Power	1275.412	1275.356	1275.446	1275.55	**1275.93**	1275.97	1275.94	1276.01	1276.03
Total losses	12.412*	12.3598*	12.446*	12.55*	**12.95**	12.98	12.9361	12.9584	13.0217
Total Cost ($/hr)	15442.67	15442.75	15443.09	15446.02	**15449.58**	15449.7	15450	15450	15459

*the computed total losses with the Kron's loss formula are higher than these values.

The summation of the total power generated of the best solution obtained by the differential evolution algorithm is 10,500 MW, and the mismatch is zero. The total time required for the proposed approach to solve the above problem as computed using (4.2) and (4.3) is 31.5037 sec, where $t_1 = t_3 = 0.92$ sec, $t_2 = 29.6637$ sec, and $t_4 = 0$ second.

4.4 Case Study 4: Comparing the Proposed Approach with Previously Proposed Approaches

This case study is divided into four parts. In part A, the proposed approach in this thesis is compared with the discrete lambda consensus approach proposed in [2]. In part B, the proposed approach is compared with the approach proposed in [1] which utilizes the mutually coupled dynamical systems for achieving consensus between the agents. In part C, the proposed approach is compared with the distributed approach proposed in [4] from the perspective of handling the transmission losses. Finally, Part D provides a comparison between the proposed approach and the centralized approach for solving the EDP.

TABLE 4.3

40-GENERATOR TEST SYSTEM: COMPARISON OF THE DE ALGORITHM SOLUTION WITH SOME OF THOSE REPORTED IN THE LITERATURE

Method	Min ($/hr)	Average ($/hr)	Max ($/hr)
IFEP [27]	122,624.35	123,382.00	125,740.63
AA [5]	121,788.70		
CPSO–SQP [28]	121,458.54	122,028.16	NA
FCASO-SQP [29]	121,456.98	122,026.21	NA
DE/BBO [30]	121,420.90	121,420.90	121,420.90
aBBOmDE [26]	121,414.87	121,487.85	121,568.32
CE-SQP [31]	121,412.88	121,423.65	NA
DE	**121,412.68**	**121,439.89**	**121,479.63**
FAPSO–VDE [32]	121,412.56	121,412.61	121,412.78
CSA [33]	121,412.54	121,520.41	121,810.25

4.4.1 Part A

In this section, the discrete lambda consensus algorithm presented in [2] is simulated for further investigation and understanding of that algorithm and for the purpose of comparing it with the proposed approach in this thesis. There are some variants of the discrete lambda consensus algorithm over the literature; however, most of them have in common the concept of locally updating the lambda variable based on the state of the lambda variables of the neighbors and a feedback related to the total power mismatch. Accordingly, the agents reach a consensus on the optimal lambda value while satisfying the constraint of total power mismatch equal to zero. One of the variants of the discrete lambda consensus algorithm is that presented in [2] which consists from the following difference equations

$$\lambda_i(k+1) = \sum_{j \in N_i^+} D_{i,j} \lambda_j(k) + \sigma \Delta P(k) \tag{4.4a}$$

$$P_i(k+1) = \beta_i \, \lambda_i(k+1) + \alpha_i \tag{4.4b}$$

$$\Delta P_i(k+1) = \sum_{j \in N_i^+} C_{i,j} \, \Delta P_j(k) - (\mathrm{P}_i(k+1) - \mathrm{P}_i(k)) \tag{4.4c}$$

Where $\lambda_j(k)$ is the local estimation of the incremental cost by generator *i*

σ is a small positive number acts as a feedback gain.

$D_{i,j}$ is the element of a row stochastic matrix associated with the assumed strongly connected and directed communication graph. It is computed as follows:

$$D_{i,j} = \begin{cases} \dfrac{1}{d_i^+} & if\ \ j \in N_i^+ \\ 0 & otherwise \end{cases} \quad \forall i, j \in V \tag{4.5}$$

Where d_i^+ is the in-degree of node *i* and equal $|N_i^+|$, where N_i^+ is the in-neighbors of the *i*-th node and $|N_i^+|$ means the cardinality of the set N_i^+.

$\Delta P_i(k)$ is the local estimation of the mismatch between the total power generation and total demand.

$P_i(k)$ is the output generation of unit *i*.

β_i, α_i are constants for each generation unit and computed from the cost coefficients of the quadratic cost function $C_i(P_i) = a_i\, P_i^2 + b_i\, P_i + c_i$ as follows:

$$\beta_i = \frac{1}{2\,a_i},\ \alpha_i = -\frac{b_i}{2\,a_i} \tag{4.6}$$

$C_{i,j}$ is the element of a column stochastic matrix associated with the assumed strongly connected and directed communication graph. It is computed as follows:

$$C_{i,j} = \begin{cases} \dfrac{1}{d_j^-} & if\ i \in N_j^- \\ 0 & otherwise \end{cases} \quad \forall i, j \in V \tag{4.7}$$

Where d_j^- is the out-degree of node *j* and equal $|N_j^-|$, where N_j^- is the out-neighbors of the *j*-th node and $|N_j^-|$ means the cardinality of the set N_j^-.

The authors in [2] assumed a directed communication graph as shown in figure 4.9. The results of simulating the above algorithm with the initial data provided in [2] are shown in figures 4.10-4.13. Figure 4.10 shows the lambda variable for the four generators, figure 4.11 shows the power generation value for the four generators, figure 4.12 displays the power mismatch estimated by the four generators and figure 4.13 compares the total power generated with the total demand.

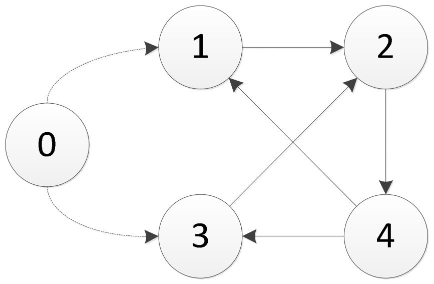


Figure 4.9 The assumed communication graph in [2]

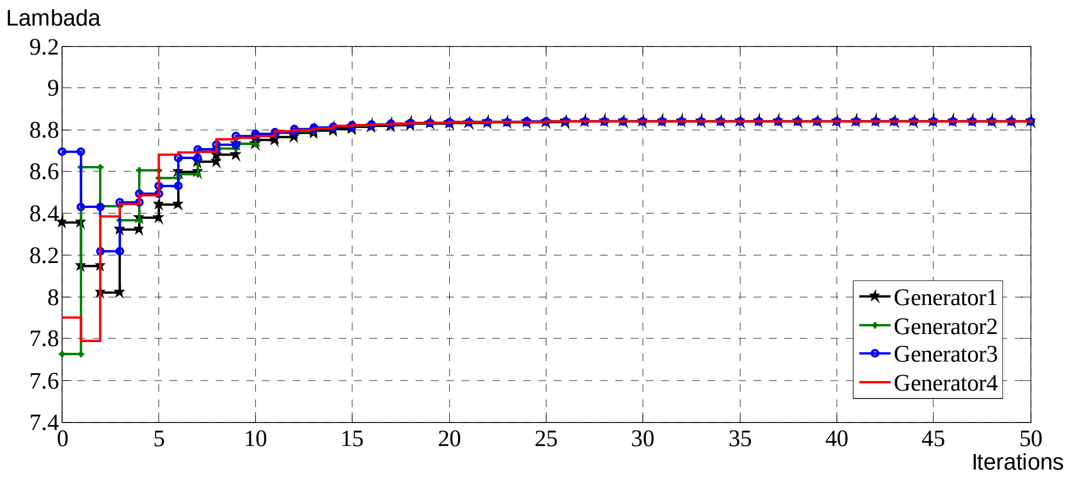


Figure 4.10 Lambda variables for the four generators

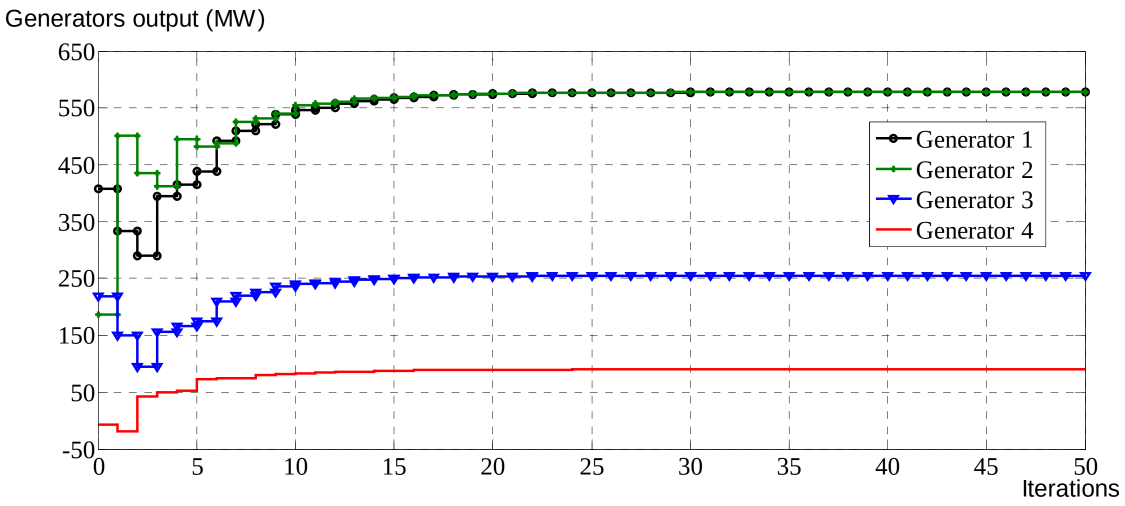


Figure 4.11 Output power generated by each generator

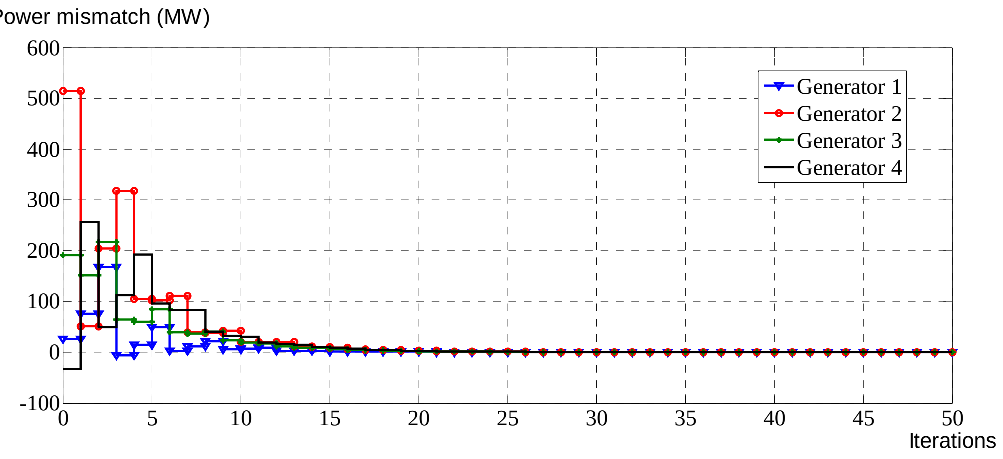


Figure 4.12 Power mismatch estimation by the four generators

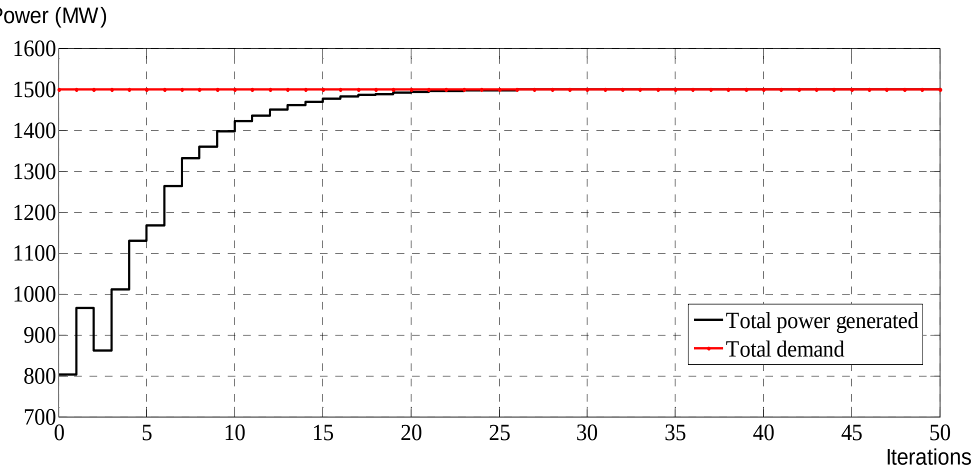


Figure 4.13 Total power generated and total demand

Now, if one of the communication links has failed like the link 4-1, then the communication graph will be as shown in figure 4.14 and the communication graph is no longer strongly connected.

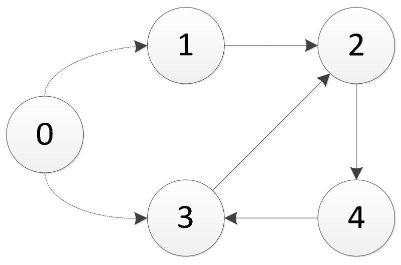


Figure 4.14 The assumed communication graph in [2] without the communication link 4-1

The results from the algorithm in this case are shown in figures 4.15-4.16. Figure 4.15 shows that there is no consensus on the lambda variable among the agents, meaning that the agents are not dispatched in an economical way. Figure 4.16 shows that there is a mismatch between the total power generated and the total demand, so the failure of link 4-1 should be accompanied by an outage for unit 1. If this does not happen, the economic dispatch for the system will fail and a large mismatch will exist between the total power generated and the total demand. The implementation of the lambda discrete consensus algorithm can be done using the multi-agent technology and the appropriate iterative communications between the agents. An iteration of the above algorithm is equivalent to a set of communication signals or messages between the agents.

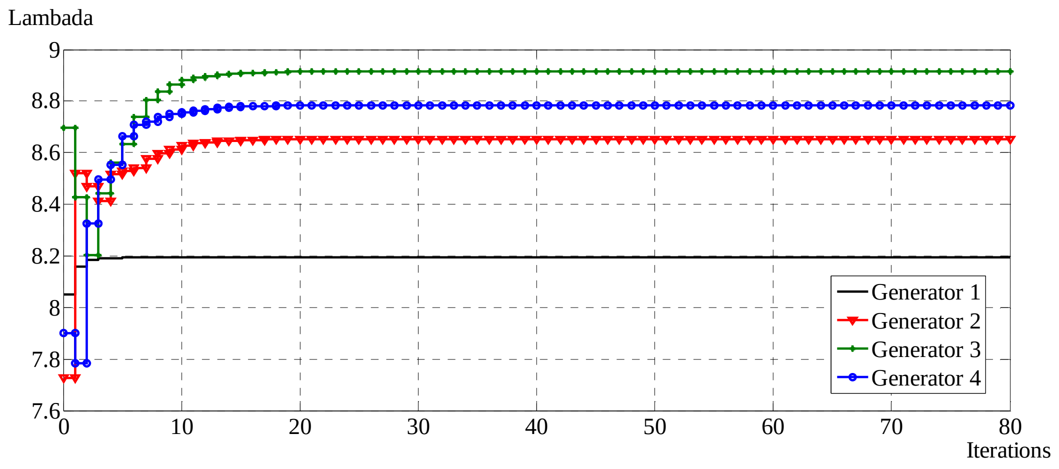


Figure 4.15 Lambda variables for the four generators

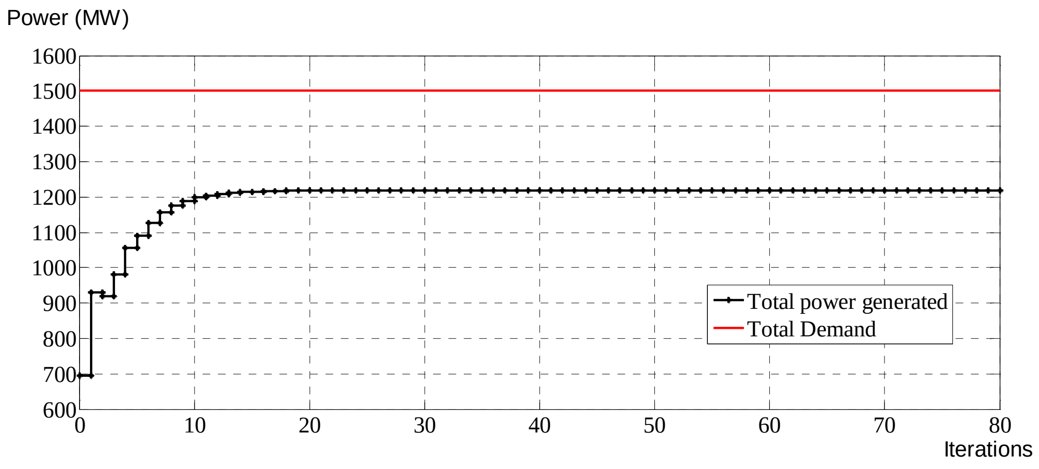


Figure 4.16 Total power generated and total demand

The approach proposed in this thesis has been applied to solve the same problem, and the equal incremental cost method has been used by each agent to solve the problem locally during stage 2. Stage 3 is not needed in this case. The problem has been solved for two cases: one with a loss of one communication link, and the second without this loss. For both cases, the agents reached consensus on the following solution: λ = 8.8397 \$/MWh, p_1 = 577.3547, p_2 = 577.3547, p_3 = 255.0741 and p_4 = 90.2165. Although the lambda consensus algorithms involve a consensus on few variables like the lambda variable and power mismatch, the number of communication iterations and messages required by the agents is higher than the number of iterations and messages required by the agents to reach consensus on the system data that are needed to fully solve the problem locally by each agent. In order to calculate the total time required by the proposed approach in [2] based on the same approximation of the communication time used in this thesis, and to compare the amount of data transferred through the communication graph in the proposed approach in [2] with those transferred through the communication graph in this thesis, the total number of messages and communication iterations used by the proposed approach in [2] have to be computed. In the algorithm proposed in [2], the iterative communication is done on the lambda variable and the total power mismatch. For the node that has in-degree equal to 1, each update to the lambda and estimated power mismatch values corresponds to receiving a message that contains two scalar numbers per iteration. If the out-degree is equal to 1, then this is equivalent to sending a message that contains 2 scalar numbers per iteration. Table 4.4 summaries the iterative communication used in the proposed algorithm in [2] over 4-bus system and using the assumed communication graph in figure 4.9.

TABLE 4.4
TOTAL MESSAGES TRANSFERRED PER ITERATION BY THE PROPOSED APPROACH IN [2]

	In-degree	Out-degree	Messages sent per iteration	Total scalars transferred per iteration
Node 1	1	1	1	2
Node 2	2	1	1	2
Node 3	1	1	1	2
Node 4	1	2	2	4
		Total	5 per iteration	10 per iteration

From figures 4.10 - 4.13, it is clear that the total number of communicative iterations is approximately 20. Therefore, the total number of messages flowing over the 4-bus system using the proposed algorithm in [2] is 5 x 20 = 100, and total number of scalar values transferred over the network is 10 x 20 = 200. On the other hand, if the above system is assumed for the approach proposed in this thesis, each message consists of a tuple that contains 8 scalar values, as follows:

$$< Bus\ number, Bus\ stamp, a_i, b_i, c_i, P_{Li}, P_i^{\min}, P_i^{\max} >$$

Where P_{Li} is the load at bus i. Each node sends two messages per iteration and receives two messages per iteration. For the four-bus system, the total messages sent per iteration equals 2 x 4 = 8, and the total number of iterations, based on (4.1), is equal to 2. With the proposed algorithm, the total number of messages that flow over the four-bus system is 2 x 8 = 16, and the total number of scalar values transferred over the network is 16 x 8 = 128.

The time required with the approach proposed in [2] is equal to the time required for 20 communication iterations, which is considered equal to the time required for sending 20 consecutive messages between two agents, or 0.92 second. On the other hand, the total time required for the proposed approach in this thesis to solve the above problem is 0.282 sec: the time required for stage 1 is 0.092 sec, and the time required for each agent to solve the problem locally using the equal incremental cost method is 0.19 sec.

To summarize, the proposed approach in this thesis can overcome the consensus on lambda approach in solving the convex economic dispatch problem with respect to the total time required and the amount of data that flow through the network, yet these are not the only advantages of the proposed approach in this thesis. Other important advantages are that the proposed approach can be applied for solving the non-convex economic dispatch problem, and it provides a suitable mechanism for incorporating the transmission losses in an accurate way.

4.4.2 Part B

In this section, the proposed approach in [1] that utilizes the mutually coupled dynamical systems for achieving consensus between the agents is studied and simulated for further investigation and for the purpose of comparing it with the proposed approach in this thesis. The limitations of this approach are discussed in more detail at the end of this section. The data of the three-generator system used in this case study exist in [1]. The continuous consensus algorithm proposed in [1] utilizes the following coupling protocol:

$$u_i = \frac{K}{c_i} \sum_{j \in N_i} f(x_j - x_i) \tag{4.8}$$

Where

K is a feedback control gain

c_i is a fixed parameter that adjusts the response of each agent compared to the other agents.

$f(x_j - x_i)$ is a function of the difference between the agent state x_i and the neighbor state x_j

N_i is the set of neighbor agents to node *i*.

And the state of the agents in the network evolves according to

$$\dot{x}_i = f(x_i, u_i) \tag{4.9}$$

Which then leads to the agent states reaching a consensus on the following value:

$$\tilde{x} = \frac{\sum_{i}^{N} c_i \, x_i(0)}{\sum_{i}^{N} c_i} \tag{4.10}$$

Where

$\tilde{x}$ is the weighted average of the initial states of the agents.

N is the total nodes in the network.

The unweighted average consensus algorithm is a special case of the above algorithm in which $c_i = c_j = K$ for all i, j$\in [1, N]$. In the case of the unweighted average consensus algorithm, the agents reach a consensus on the following value:

$$\tilde{x} = \frac{1}{N} \sum_{i=1}^{N} x_i(0) \tag{4.11}$$

In [1], a mix of the above weighted and unweighted average consensus algorithms has been used in solving the economic dispatch problem as follows:

Step 1) the agents compute the total system demand using the unweighted average consensus algorithm in which the initial state of each agent is

$$x_i(0) = N \times P_{D,i} \quad \forall i \in [1, N] \tag{4.12}$$

Where $P_{D,i}$ is the load at bus *i*, and then the agents will reach consensus on the following value

$$\tilde{x} = \sum_{i=1}^{N} P_{D,i} \tag{4.13}$$

Considering the 3-generation unit system in [1], and assuming that the total system load is equal to 975 MW and distributed between these three buses as follows, $P_{D,1} = 200, P_{D,1} = 300, P_{D,3} = 475$, then the agents reach a consensus on the total system load as shown in figure 4.17.

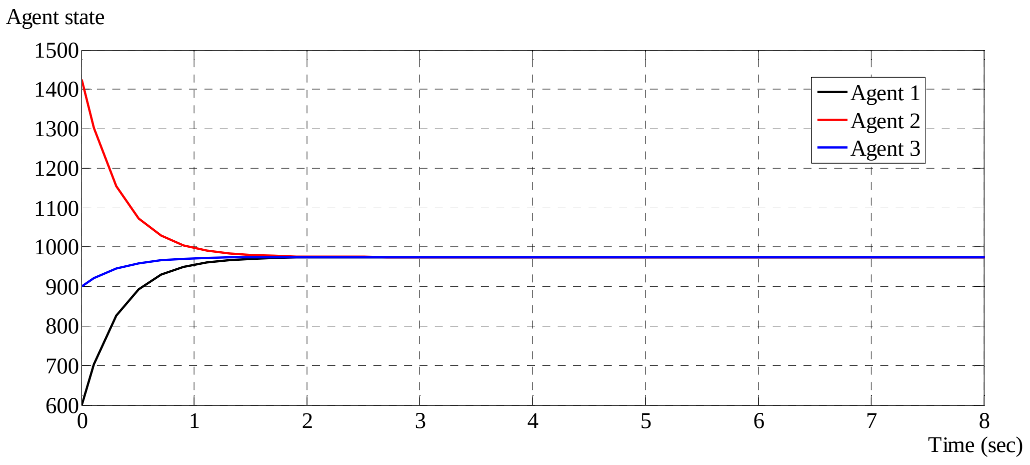


Figure 4.17 Total system load as computed by each agent

Step 2) the agents use the weighted average consensus algorithm to reach consensus on the global variable $\lambda - \Delta\lambda$ through initializing the agent states using the following formula

$$x_i(0) = \left.\frac{dC_i}{dP_i}\right|_{x_i} + \frac{\Delta P}{n_g} \left.\frac{d^2C_i}{dP_i^2}\right|_{x_i} \tag{4.14}$$

And each agent adapts the following weight

$$c_i = \frac{1}{\left.\frac{d^2C_i}{dP_i^2}\right|_{x_i}} \tag{4.15}$$

Where

$\left.\frac{dC_i}{dP_i}\right|_{x_i}$, $\left.\frac{d^2C_i}{dP_i^2}\right|_{x_i}$ are the first and second derivative of the quadratic cost function $C_i(P_i) = a_i\, P_i^2 + b_i\, P_i + c_i$ evaluated at x_i .

n_g is the total number of generating units.

ΔP is the total power mismatch and it is equal to $\sum_{i=1}^{N} P_{D,i}$ in the first iteration.

By considering the 3-generator system again, the consensus on the global variable $\lambda - \Delta\lambda$ is shown in figure 4.18 where the final value of $\lambda - \Delta\lambda = 9.16315$

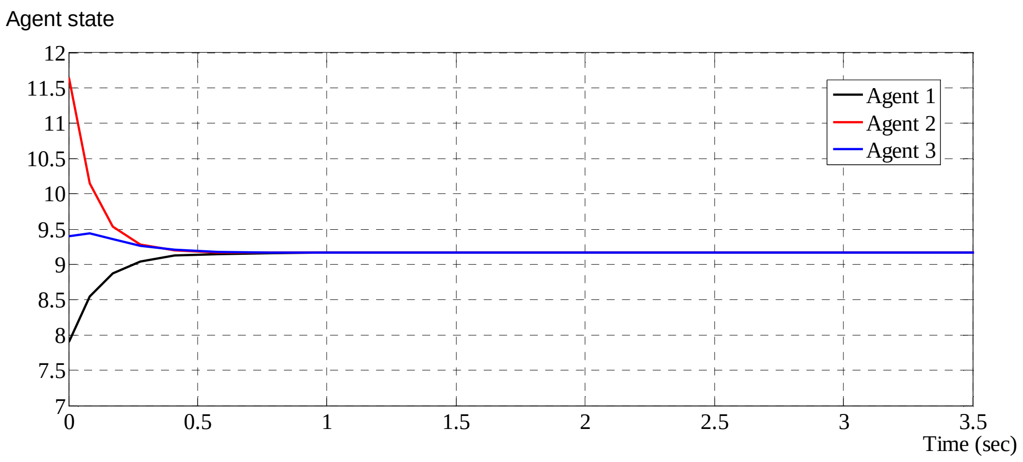


Figure 4.18 Consensus on the global variable $\lambda - \Delta\lambda$ by the agents

Step 3) after the agents reach consensus on the value of $\lambda - \Delta\lambda$, each agent uses this value to update its solution as follow

$$\Delta P_i = \frac{\left.\frac{dC_i}{dP_i}\right|_{x_i} + \Delta\lambda - \lambda}{\left.\frac{d^2C_i}{dP_i^2}\right|_{x_i}} , \quad P_i := P_i - \Delta P_i \qquad \forall i \in [1, n_g] \tag{4.16}$$

In the first iteration, P_i is assumed to equal zero.

Step 4) each agent checks for constraints violation and decides if it will continue participating or not. If the agent solution violates either the upper or lower generation limit, the agent adjusts its solution to that violated limit, and does not continue in the dispatching process.

In the case of the three-generator system, the first generator reaches its upper limit, which is 450 MW. This generator will then fix its generated output power at this limit and will not continue in the dispatching process. The output power from the second and the third units are $p_2^{(1)} = 305.3,\ p_3^{(1)} = 186.8$ MW.

Step 5) the agents use the unweighted average consensus algorithm to compute the total mismatch. Each agent initializes its state using the following formula:

$$x_i(0) = N \times (P_{D,i} - P_i) \quad \forall i \in [1, N] \tag{4.17}$$

Then, if there is still a mismatch, the agents continue from step 2.

In the case of the three-generator system, the total mismatch is -32.9 MW due to the limitation enforcement at generator 1. Therefore, the agents of generator two and three will continue from step two to recover this mismatch in an economical way. Figure 4.19 shows the new computed value of $\lambda - \Delta\lambda$, which is equal to 9.4.

After reaching consensus on the $\lambda - \Delta\lambda$ value, each agent will compute the corresponding update to its output power as in step 3 and the final solution to the 3- generators dispatching problem will be

$$p_1^{(2)} = 450\,,\ p_2^{(2)} = 325 \text{ and } p_3^{(2)} = 200$$

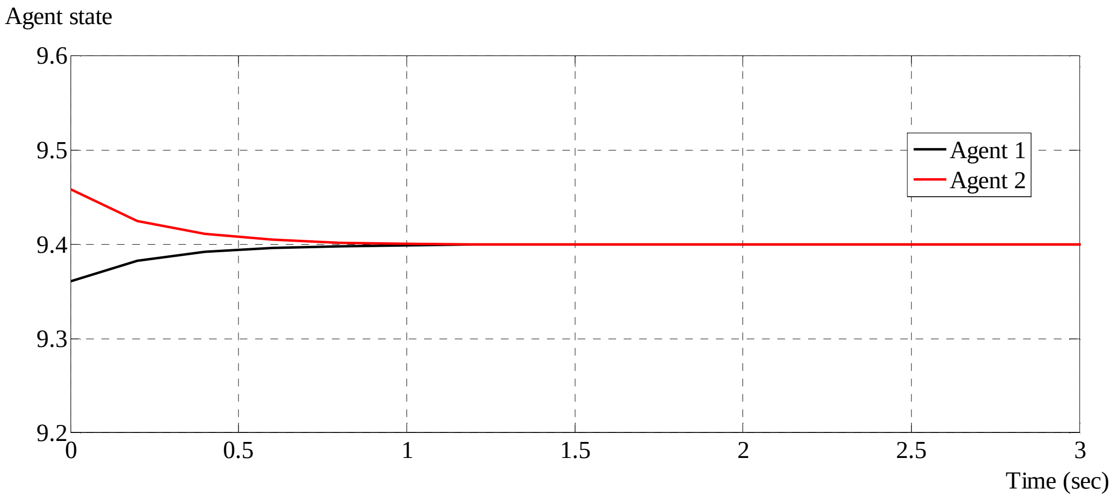


Figure 4.19 Consensus on the global variable $\lambda - \Delta\lambda$ by the agents

Implementing the continuous consensus algorithm involves equipping each agent with an oscillator (dynamical system). Considering an implementation based on a continuous state, the simplest dynamical system is a first-order dynamic circuit. This implementation can be achieved by augmenting each agent with a capacitor. These capacitors and the small resistance of the lines that connects the capacitors form a mutually coupled dynamical system. If each capacitor has been charged by a voltage corresponding to the agent's initial value and these capacitors have been connected together over a network, the agents will reach a consensus on the average value of the initial agents' states. This implementation of mutually coupled dynamical systems is based on continuous states. Figure 4.20 shows the simulation of this concept in Matlab and figure 4.21 shows that the voltages across all the capacitors have reached the same value, which is equal to the average of the agents' initial values.

The idea of mutually coupled dynamical systems is very interesting. At the beginning of this thesis work, the approach proposed in [1] was chosen for further improvement of that approach; however, this process was not continued due to the disadvantages discussed in the following paragraphs, and a new approach has been proposed in this thesis to recover these disadvantages.

The possible continuous state based implementation discussed above has the disadvantage of additional capacitors and controllable switches at each node. The application of this implementation requires investigating any possible problems that may result from the application of these circuits. Such possible problems include the need for multiple and continuous discharging and charging of the capacitors and losses in the lines connecting the capacitors. The last issue may reduce the accuracy of information sharing between the nodes. For these reasons, implementing this approach using capacitors and switches is not expected even though being based on simple concept. It is expected that an implementation based on a discrete version of this algorithm will be used in which the agents will share their information and reach a consensus using communication messages. The corresponding discrete version of this algorithm can be implemented using the multi-agent technology; however, it is expected that the number of communication messages and iterations will not be less than that used by the approach discussed in the previous case study, and the total communication time delay will not be lower. This expectation is due to the fact that the approach proposed in [1] requires the application of a series of unweighted and weighted consensus algorithms for solving the convex economic dispatch problem instead of just the application of one consensus algorithm or two consensus algorithms running in parallel as proposed in [4].

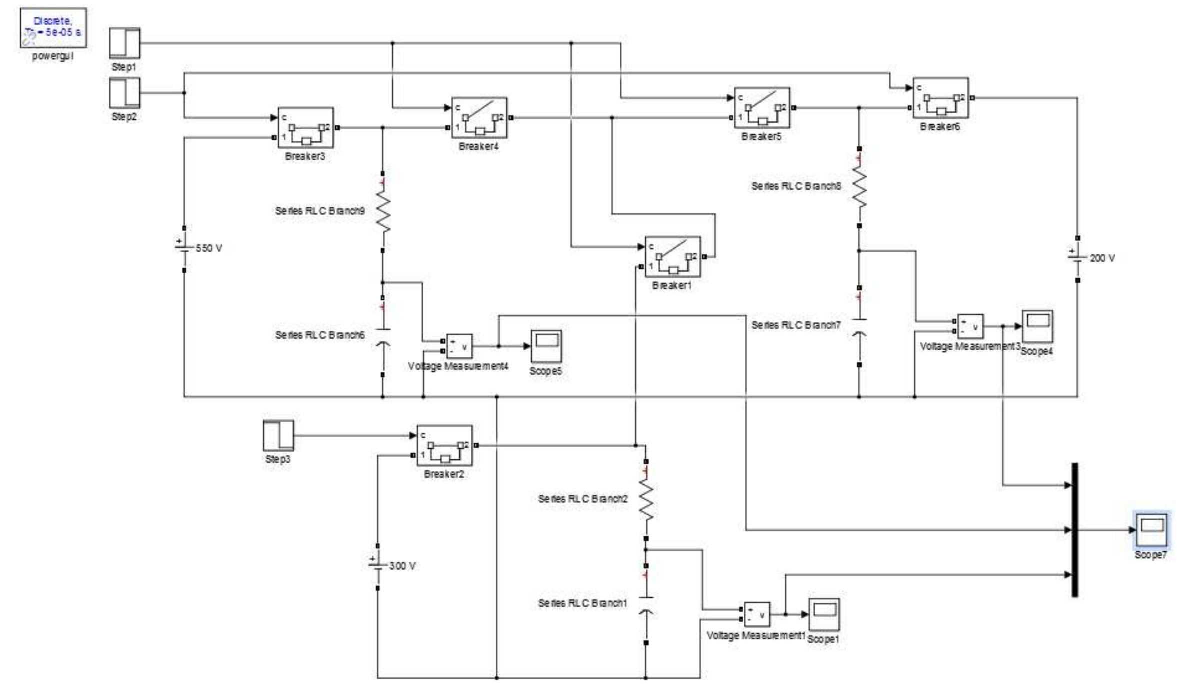


Figure 4.20 Simulation of a possible implementation to the continuous consensus algorithm in Matlab

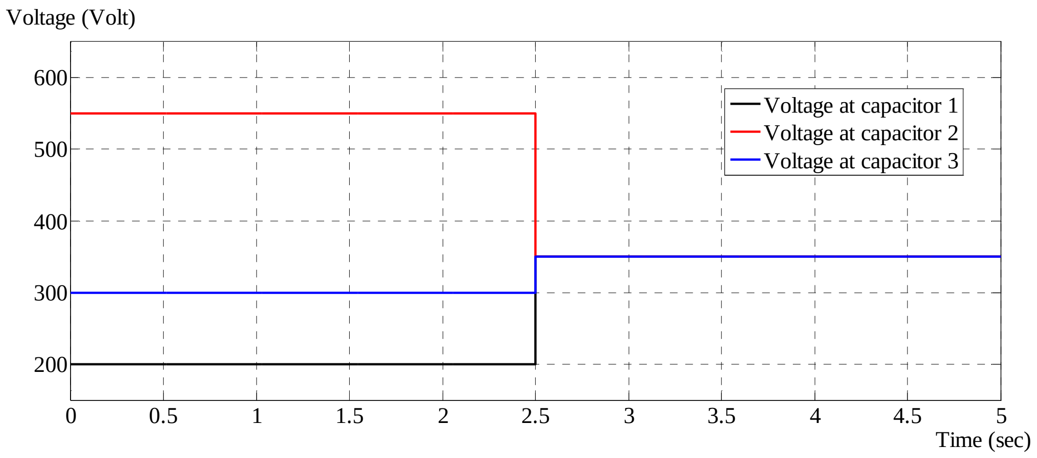


Figure 4.21 Voltages of the capacitors in Figure 4.20.

The above concerns related to implementing the proposed approach in [1] are not the main disadvantages of this approach. As noted in equation 4.12, this approach requires initializing the state of each agent with a value proportional to the number of agents. Consequently, each agent has to know this number. Knowing this number requires central authority assistance; defining the losses coefficients of the network for each agent also requires central authority assistance. For this reason, this approach is a decentralized approach and not a fully decentralized approach since it still preserves the need for central authority for completing its task successfully. Therefore, the approach proposed in [1] lacks the suitable mechanism for incorporating the transmission losses in a fully decentralized manner; in addition, this approach cannot be extended to solve the non-convex economic dispatch problem if required.

4.4.3 Part C

In [4], the authors proposed a technique for incorporating transmission losses while solving the EDP using the consensus on lambda approach. They used a six-bus, three- generator, and 11-line system, the data for which is available in [34]. They also doubled the system load from 210 MW to 420 MW and applied their proposed algorithm in order to dispatch the system at the 420 MW load. The results obtained by this proposed algorithm was $p_1 = 127.33$ MW, $p_2 = 151.48$ MW, and $p_3 = 148$ MW [4]. According to [4], with this dispatching, the power mismatch becomes zero, and the generation-demand equality constraint is satisfied; however, summing the output power of the three generators gives a total power generated of 426.81 MW, which means that the total transmission losses are 6.81 MW. When the power flow problem has been solved with the new load values and output generator power, the total transmission losses are found to be 19.199 MW. The proposed approach in this thesis has been applied to solve the economic dispatch problem for that system. In stage 2, the penalty factors have been used with the equal incremental cost method for incorporating the transmission losses [19], and there is no need to stage 3. To handle the significant change in load from one dispatching period to the next, the following strategy is applied. Each agent solves the EDP for a total system load of 420 MW, without the inclusion of the transmission losses. The power levels output from the units are then used as initial values for the power flow problem in order to compute the loss coefficients, following which, each agent uses the equal incremental cost method with the penalty factors to resolve the EDP with the inclusion of the transmission losses. The output results from dispatching the units are shown in Table 4.5.

TABLE 4.5
RESULTS FOR THREE-GENERATOR SYSTEM WITH CONVEX COST FUNCTIONS

Output power	Output power	Output power
P_1 139.1997	P_2 153.8351	P_3 146.2829
Total power 439.3177	Total losses 19.3177	Total cost 5923.91 $/h

When the power flow problem is solved with the output power indicated in Table 4.5, the total transmission losses are found to be 19.2 MW. As with the first case study in [4], this case study does not consider the generation limits.

4.4.4 Part D

This section provides a comparison between the new decentralized approach and the centralized approach with respect to solving the EDP. Although more data flow over the communication network than in a centralized system, the decentralized approach has the advantage of no single system node is subject to a communication bottleneck or increased management complexity relative to the other system nodes. In centralized systems, if one node is assumed to be distinct from the other system nodes so that it provides central management for the system, as shown in Figure 4.22 (a), and if it is assumed that the central node needs to collect data from the other system nodes, the central node then solves the EDP and sends the results back to the other system nodes.

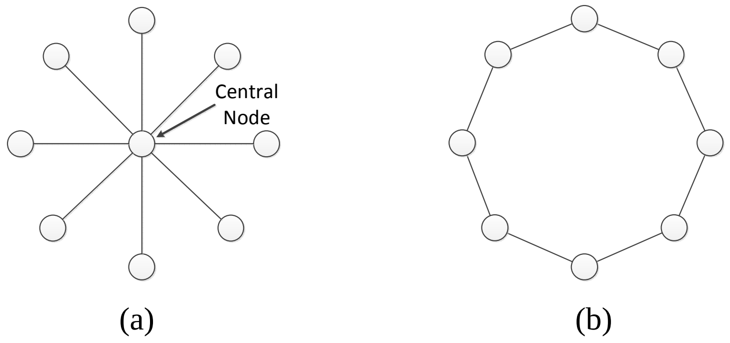


Figure 4.22 Case study 4, part d: (a) star topology for centralized system; (b) ring topology for decentralized system.

If the total number of nodes excluding the central node is n, then the total number of messages communicated for the centralized system is 2n. All of these messages are either received or sent by the central node for the performance of one task such as solving the EDP. As well, n communication links end at the central node. On the other hand, if a fully decentralized operation with ring communication graph is assumed, as shown in Figure 4.22 (b), each node has only two communication channels connected to it whatever the size of the system. Based on the assumption, as stipulated in the proposed decentralized approach, that each node needs to collect system data in order to process these data locally and produce a decision with the same efficiency as that in the centralized system, the number of messages then received by each node in the system equals two per iteration. The total number of iterations required for sharing the information between the agents is proportional to the system size and can be computed according to (4.1). The increased number of communication iterations adds a time delay; however, as shown in the previous case studies, this time delay varies from fractions of seconds in small systems to approximately 1.84 sec in the 40-unit system, which means that the time delay is negligible compared to the dispatching period which has a time scale from 5 min to a few hours.

Till this point, the centralized system and the proposed decentralized system have been compared with respect to the communication network. An important advantage that the proposed decentralized system has is that there is no leader or centralized node and hence no concerns related to single point failure issue [1]-[6]. Also, decentralized systems are more scalable and more flexible with respect to system changes than centralized systems [4].

Chapter 5
Discussion and Conclusion

5.1 Discussion and Future Work

Although this thesis has demonstrated several advantages associated with the proposed fully decentralized approach for solving the EDP, this section discusses possible limitations associated with the practical application of the proposed approach in real-life systems and suggests future research that could be expected to address these limitations. During the process of tuning the differential evolution algorithm, it was noted that the differential evolution algorithm control parameters are sensitive to the formulation and dimensions of the problem. If the differential evolution algorithm has been used to solve other problems with different dimensions based on the same parameter settings described in this thesis, it may therefore not provide a high-quality or satisfactory solution. This problem is a general problem associated with many metaheuristic techniques and with any centralized, decentralized, or distributed approach that utilizes such techniques. This problem can be addressed through the use of a more efficient metaheuristic technique that either relies on control parameters less sensitive to the problem formulation and dimensions or incorporates a self-tuning feature. Examples of already-existing self-tuning differential evolution algorithms are those reported in [32] and [35].

Other challenges that may be associated with the practical implementation of the proposed approach are related to cyber-attacks. Sharing system and unit data between system nodes is not an issue because these data can be encrypted if necessary during their transfer between system nodes; however, injecting false data that will misdirect the system operation may be considered problematic. This drawback is associated with any distributed and decentralized approach, including the consensus on lambda approach, and can occur in any system that employs a communication layer for sharing data among system nodes because unauthorized individuals are able to access these data or communication networks. For this reason, although this research has included the successful use of the internet for creating a communication link between two nodes at different locations, it is expected that a practical implementation of the proposed approach would require a special dedicated communication network for connecting the system nodes, so that use of the internet could be avoided.

Future work can include the extension of the proposed approach to solve the dynamic economic dispatch problem over a 24-hour time horizon. This includes adapting the proposed fully decentralized approach for solving the unit commitment problem. In this case, the system and units data sharing process through the flooding-based consensus algorithm can be applied once per day before solving the unit commitment problem. Thereafter, the flooding-based consensus algorithm will be run for each dispatching period in order to update the agents about the system load. If an emergent change in the network state has been detected by any agent, then this agent will initiate the flooding-based consensus algorithm to share its information about this change in order to the agents can take appropriate action if needed. Future work can also include studying different communication graph topologies and the effect of the variability of the network topology on the proposed approach operation.

5.2 Conclusion

This thesis has presented a new fully decentralized approach for solving the EDP based on data networks and multi-agent technology. For sharing the information required by the agents to act intelligently, a flooding-based consensus algorithm over a ring communication graph that connects the agents has been proposed. With the current advances of metaheuristic techniques and the application of the proposed approach, the non-convex economic dispatch problem can be solved efficiently with high quality solutions in a fully decentralized manner. For solving the non-convex economic dispatch problem, the proposed approach consists of three stages: the flooding-based consensus algorithm is used for sharing the units and system data during the first stage; a suitable metaheuristic technique is employed in the second stage for solving the NCEDP; and the flooding-based consensus algorithm is reapplied in the third stage for sharing the information required. The same approach can be adapted to solve the EDP with convex cost functions through using the equal incremental cost method in the second stage and the third stage is not needed. The proposed approach and its methodology of sharing the data directly between the system nodes offers significant advantages such as incorporating the transmission losses accurately and solving the NCEDP in a fully decentralized manner. An experimental setup has been employed for approximating the communication time required with the proposed approach. The time delay introduced by the operation of the proposed approach varies from fractions of seconds in small systems to approximately 1.84 seconds in the 40-unit system, which is negligible compared to the dispatching period which has a time scale from 5 min to a few hours. Also, this time delay is smaller than that introduced by the well-known consensus on lambda approach for solving the EDP with convex cost functions as demonstrated in one of the case studies presented in this thesis. The results obtained from the four case studies examined in this thesis indicate that the significant advantages of the proposed approach candidate it for solving other smart grid decision making problems in a fully decentralized manner. Future work could explore solving the security constrained economic dispatch problem, optimal power flow problem, unit commitment problem, and any other challenges that require information sharing between the system nodes in order to be implemented in a fully decentralized manner.

Appendix A

The following Table provides the output power of the units which represents the best solution computed by the differential evolution algorithm under the following control parameters

- Number of runs: 120
- No of iterations: 1,500
- Scaling factor = 0.09
- Crossover probability = 0.5
- Population size = 500

Table A1
40-GENERATOR TEST SYSTEM: OUTPUT POWER FROM THE UNITS FOR THE BEST SOLUTION

Power output (MW)		**Power output (MW)**		**Power output (MW)**		**Power output (MW)**	
P_1	110.7997	P_{11}	94.0003	P_{21}	523.2794	P_{31}	189.9991
P_2	110.8000	P_{12}	94.0001	P_{22}	523.2793	P_{32}	189.9999
P_3	97.4011	P_{13}	214.7593	P_{23}	523.2792	P_{33}	189.9999
P_4	179.7331	P_{14}	394.2790	P_{24}	523.2794	P_{34}	164.8001
P_5	87.8003	P_{15}	394.2795	P_{25}	523.2793	P_{35}	199.9660
P_6	139.9999	P_{16}	394.2792	P_{26}	523.2806	P_{36}	194.4307
P_7	259.5998	P_{17}	489.2799	P_{27}	10.0001	P_{37}	109.9998
P_8	284.5998	P_{18}	489.2791	P_{28}	10.0001	P_{38}	109.9997
P_9	284.5998	P_{19}	511.2792	P_{29}	10.0000	P_{39}	109.9991
P_{10}	130.0001	P_{20}	511.2795	P_{30}	87.7999	P_{40}	511.2796
Total power		**10,500 MW**		**Total cost**		**121412.6811 $/hr**	

Bibliography

[1] V. Loia and A. Vaccaro, “Decentralized economic dispatch in smart grids by self-organizing dynamic agents,” IEEE Transactions on Systems, Man & Cybernetics: Systems, vol. 44, no. 4, pp. 397-408, 2014.

[2] Shiping Yang, Sicong Tan, and Jian-Xin Xu, “Consensus Based Approach for Economic Dispatch Problem in a Smart Grid,” IEEE Transactions on Power Systems, vol. 28, no. 4, pp. 4416 - 4426, 2013.

[3] Z. Zhang and M.-Y. Chow, “Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid,” IEEE Transactions on Power Systems, vol. 27, no. 4, pp. 1761–1768, 2012.

[4] G. Binetti, A. Davoudi, F.L. Lewis, D. Naso, and B. Turchiano, “Distributed Consensus-Based Economic Dispatch With Transmission Losses,” IEEE Transactions on Power Systems, This article has been accepted for inclusion in a future issue of the journal.

[5] G. Binetti, A. Davoudi, D. Naso, B. Turchiano, and F. Lewis, ”A Distributed Auction-Based Algorithm for the Non-Convex Economic Dispatch Problem,” IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1124–1132, 2014.

[6] W. Zhang, W. Liu, X. Wang, L. Liu, and F. Ferrese, ”Online Optimal Generation Control Based on Constrained Distributed Gradient Algorithm,” IEEE Transactions on Power Systems, This article has been accepted for inclusion in a future issue of the journal.

[7] D. C. Walters and G. B. Sheble, “Genetic algorithm solution of economic dispatch with valve-point loading,” IEEE Transactions on Power Systems, vol. 8, no. 3, pp. 1325–1332, Aug. 1993.

[8] F. L. Bellifemine, G. Caire, and D. Greenwood, Developing Multi-Agent Systems With JADE. Hoboken, NJ: Wiley, 2007.

[9] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” Proc. IEEE, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[10] A. Tahbaz-Salehi and A. Jadbabaie, “A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times,” in Proc. 45th IEEE Conference on Decision & Control, Dec. 2006, pp. 4664–4669.

[11] G. Binetti, M. I. Abouheaf, F. L. Lewis, D. Naso, A. Davoudi, and B. Turchiano, “Distributed solution for the economic dispatch problem,” in Proc. of 21st Mediterranean Conference on Control and Automation (MED), pp. 243–250), 2013.

[12] Z. Zhang, X. Ying, and M.Y. Chow, “Decentralizing the economic dispatch problem using a two-level incremental cost consensus algorithm in a smart grid environment,” in Proc. of 43th Annual North American Power Symposium, pp. 1–7, 2011.

[13] X. Ying and M.-Y. Chow, “Sampling rate selection influences on incremental cost consensus algorithm in decentralized economic dispatch,” in IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 1410–1415, 2012.

[14] Z. Zhang and M.-Y. Chow, “The leader election criterion for decentralized economic dispatch using incremental cost consensus algorithm”, in IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, pp. 2730 –2735, 2011.

[15] V. Saligrama and M. Alanyali, “A token-based approach for distributed computation in sensor networks,” Selected Topics in Signal Processing, IEEE Journal of, vol. 5, no. 4, pp. 817 –832, aug. 2011.

[16] D. P. Bertsekas and R. Gallager, Data Networks. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[17] T. Niknam and F. Golestaneh, “Enhanced bee swarm optimization algorithm for dynamic economic dispatch,” IEEE systems journal, vol. 7, no. 4, pp. 754 - 762, 2013.

[18] X. Yuan, L. Wang, Y. Yuan, Y. Zhang, B. Cao, and B. Yang, “A modified differential evolution approach for dynamic economic dispatch with valve-point effects,” Energy Conversion and Management, vol. 49, no. 12, pp. 3447–3453, Dec. 2008.

[19] H. Saadat, Power System Analysis, 2nd ed. New York: McGraw-Hill, 2002.

[20] Z. Gaing, “Particle swarm optimization to solving the economic dispatch considering the generator constraints,” IEEE Transactions on Power Systems, vol. 18, no. 3, pp. 1187–1195, Aug. 2003.

[21] I. Selvakumar and K. Thanushkodi, “A new particle swarm optimization solution to non-convex economic dispatch problems,” IEEE Transactions on Power Systems, vol. 22, no. 1, pp. 42–51, Feb. 2007.

[22] I. Ciornei and E. Kyriakide, “A GA-API solution for the economic dispatch of generation in power system operation,” IEEE Transactions on Power Systems, vol. 27, no. 1, pp. 233–242, Feb. 2012.

[23] K. T. Chaturvedi, M. Pandit, and L. Srivastava, “Self-organizing hierarchical particle swarm optimization for non-convex economic dispatch,” IEEE Transactions on Power Systems, vol. 23, no. 3, pp. 1079–1087, Aug. 2008.

[24] A. Bhattacharya and P. K. Chattopadhyay, “Biogeography-based optimization for different economic load dispatch problems,” IEEE Transactions on Power Systems, vol. 25, no. 2, pp. 1064–1077, May 2010.

[25] Jun Sun, Vasile Palade, Xiao-Jun Wu, Wei Fang, and Zhenyu Wang,” Solving the Power Economic Dispatch Problem With Generator Constraints by Random Drift Particle Swarm Optimization,” IEEE Transactions on Industrial Informatics, Vol. 10, no. 1, pp. 222-232, Feb. 2014.

[26] Lohokare, M.R., Panigrahi, B.K., Pattnaik, S.S., Devi, S., Mohapatra, A., "Neighborhood search-driven accelerated biogeography-based optimization for optimal load dispatch," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 42, no. 5, pp. 641-652, 2012.

[27] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," IEEE Transactions on Evolutionary Computation, vol. 7, no. 1, pp. 83–94, Feb. 2003.

[28] Cai, J., Li, Q., Li, L., Peng, H., Yang, Y., "A hybrid CPSO-SQP method for economic dispatch considering the valve-point effects", Energy Conversion and Management, Vol. 53, no. 1, pp. 175–181, 2012.

[29] J. Cai, Q. Li, L. Li, H. Peng, and Y. Yang, "A hybrid FCASO–SQP method for solving the economic dispatch problems with valve-point effects," Energy, vol. 38, no. 1, pp. 346–353, Feb. 2012.

[30] A. Bhattacharya and P. K. Chattopadhyay, "Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch," IEEE Transactions on Power Systems, vol. 25, no. 4, pp. 1955–1964, Nov. 2010.

[31] M.S.P. Subathra, S.E. Selvan, T.A.A. Victoire, A.H. Christinal, and U. Amato, "A Hybrid With Cross-Entropy Method and Sequential Quadratic Programming to Solve Economic Load Dispatch Problem," IEEE Transactions on Power Systems, This article has been accepted for inclusion in a future issue of the journal.

[32] Taher Niknam, Hasan Doagou Mojarrad, Hamed Zeinoddini Meymand, "A Novel Hybrid Particle Swarm Optimization for Economic Dispatch with Valve-point Loading Effects", Energy Conversion and Management, vol. 52, no. 4, pp. 1800–1809, 2012.

[33] D.N. Vo, P. Schegner, W. Ongsakul, "Cuckoo search algorithm for non-convex economic dispatch," IET Generation, Transmission & Distribution, vol. 7, no. 6, pp. 645–654, 2013.

[34] A. J. Wood and B. F. Wollenberg, Power Generation, Operation, and Control. New York, NY, USA: Wiley, 1996.

[35] Y. Lu, J. Zhou, H. Qin, Y. Wang, and Y. Zhang, "Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects," Engineering Applications of Artificial Intelligence, vol. 24, no. 12, pp. 378–387, Feb. 2011.