2013-01-01

Development Of A New Genetic Algorithm To Solve The Feedstock Scheduling Problem In An Anaerobic Digester

Ana Cram
University of Texas at El Paso, accram@miners.utep.edu

DEVELOPMENT OF A NEW GENETIC ALGORITHM TO SOLVE THE FEEDSTOCK SCHEDULING PROBLEM IN AN ANAEROBIC DIGESTER

ANA CATALINA CRAM

Department of Industrial, Manufacturing & Systems Engineering

APPROVED:

Jose Espiritu Ph.D., Chair

Heidi Taboada, Ph.D.

Juan Noveron, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

Dedication

This thesis is dedicated to my husband and sons for all their support, love and trust.

DEVELOPMENT OF A NEW GENETIC ALGORITHM TO SOLVE THE FEEDSTOCK SCHEDULING PROBLEM IN AN ANAEROBIC DIGESTER

by

ANA CATALINA CRAM, Bachelors' Degree in Industrial Engineering

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Industrial, Manufacturing & Systems Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2013

Acknowledgements

First, I would like to express my sincere gratitude to my advisor Dr. Jose Espiritu for giving me the opportunity to be part of his research team, for the constant support, patience, motivation, and immense knowledge.

I would also like to thank the rest of my thesis committee: Dr. Heidi Taboada and Dr. Juan Noveron, for their encouragement, questions and comments.

My sincere thanks also go to Oswaldo Aguirre PhD. Student, for dedicating his time to help me so patiently.

Abstract

As worldwide environmental awareness grow, alternative sources of energy have become important to mitigate climate change. Biogas in particular reduces greenhouse gas emissions that contribute to global warming and has the potential of providing 25% of the annual demand for natural gas in the U.S. In 2011, 55,000 metric tons of methane emissions were reduced and 301 metric tons of carbon dioxide emissions were avoided through the use of biogas alone. Biogas is produced by anaerobic digestion through the fermentation of organic material. It is mainly composed of methane with a rage of 50 to 80% in its concentration. Carbon dioxide covers 20 to 50% and small amounts of hydrogen, carbon monoxide and nitrogen.

The biogas production systems are anaerobic digestion facilities and the optimal operation of an anaerobic digester requires the scheduling of all batches from multiple feedstocks during a specific time horizon. The availability times, biomass quantities, biogas production rates and storage decay rates must all be taken into account for maximal biogas production to be achieved during the planning horizon. Little work has been done to optimize the scheduling of different types of feedstock in anaerobic digestion facilities to maximize the total biogas produced by these systems. Therefore, in the present thesis, a new genetic algorithm is developed with the main objective of obtaining the optimal sequence in which different feedstocks will be processed and the optimal time to allocate to each feedstock in the digester with the main objective of maximizing the production of biogas considering different types of feedstocks, arrival times and decay rates. Moreover, all batches need to be processed in the digester in a specified time with the restriction that only one batch can be processed at a time. The developed algorithm is applied to 3 different examples and a comparison with results obtained in previous studies is presented.

Table of Contents

List of Tables

List of Figures

List of Illustrations

Chapter 1: Introduction

1.1 Introduction

In the US, about 9% of the greenhouse gas emissions from human activity come from Methane and globally over 60%, causing the concentration levels of this gas to rise in the atmosphere enhancing the well known greenhouse effect. Biogas from livestock manure and other renewable sources through anaerobic digestion directly reduces methane emission that otherwise would escape through natural decomposition of manure and reduces more than 1 million metric tons of carbon dioxide emissions. In the U.S. 55,000 metric tons of methane emissions are reduced and 301 metric tons of carbon dioxide emissions are avoided through the use of biogas from anaerobic digesters. Furthermore, biogas is a renewable energy source that can be combusted to create power with 41% electrical efficiency can replace natural gas and can be used as a vehicle fuel with greater efficiency than conventional fuels.

1.2 Research motivation

Comparing the effect between methane (CH_4) and carbon dioxide (CO_2) on climate change over the last century, researchers have found that the impact of CH_4 is 20 times greater than CO_2. Even though methane's lifetime is shorter than carbon dioxide as it is naturally removed from the atmosphere it traps more radiation making the globe temperatures to rise faster. While methane is emitted through natural processes, human activities are also contributing these emissions. For instance, the storage of animal's manure produces high levels of CH_4 emissions. Therefore, animals raised for food produce CH_4 emissions through their digestive process, making agriculture the primary cause of methane emissions in the world.

This thesis is motivated by the urgent need of reducing the greenhouse gas concentration levels in the atmosphere that contribute to global warming and biogas, through anaerobic digestion, offers a significant potential to lower some emissions derived by human activity.

1.3 Research objective

The main objective of this thesis is to develop a new genetic algorithm to obtain the optimal sequence in which different feedstocks will be processed and the optimal time to allocate to each feedstock in the digester in order to maximizing the production of biogas considering different types of feedstocks, arrival times and decay rates. Moreover, all batches need to be processed in the digester in a

specified time with the restriction that only one batch can be processed at a time. The developed algorithm will be applied to 3 different examples and a comparison with results obtained in previous studies will be presented.

1.4 Thesis overview

This thesis proposes the use of a Genetic Algorithm to determine the optimal sequence of batches considering different arrival times and their optimal residence times in order to maximize the biogas produced in one anaerobic digester.

The rest of this work contains the following:

Chapter 2 explains the main aspects about global warming and provides recorded global temperatures throughout the last century. It also explains in more detail how biogas can help reduce greenhouse gases emissions. In addition, information regarding the anaerobic digestion process is provided and the US status concerning this topic.

Chapter 3 provides a general description on some optimization methods that can be implemented to solve the scheduling of anaerobic digestion systems in order to maximize the production of biogas.

Chapter 4 provides literature review that approaches the scheduling of anaerobic digesters as well as the description of the problem and the step by step methodology employed to solve it. Three case studies are presented and one is compared to a previous approach.

Chapter 5 provides two sensitivity analyses for the case studies as well as a design of experiments to obtain the GA's optimal parameters in order to obtain solutions with higher gas production

Finally, chapter 6 presents the conclusions and future work.

Chapter 2: Global Warming

Chapter 2 will review the main aspects of global warming and how human activities are contributing to climate change. Some global temperature reports will be presented to provide an overview on the increments seen in the last century. A special focus on the greenhouse gas methane and the importance of reducing these emissions is going to be taken. Also, this chapter provides with information on how these emissions can be reduced through the production and use of renewable energy.

2.1 Global Warming

Global warming is the sustained increase in the average temperature of the atmosphere sufficient to cause global climate changes. Through history, the earth has undergone many periods of temperature changes, and today it is suffering one. The increased global temperatures we are experiencing now are mainly attributed to the greenhouse effect and the raised of greenhouse gases emitted by human activity such as industries and agriculture.

The intergovernmental panel on climate change (IPCC) is the most important international body in charge of providing the world with the latest knowledge in climate change and its potential impacts through scientific assessments. Although the IPCC does not perform its own research, nor does it examines data that contributes climate change, it publishes special reports on climate change with the help of thousands of scientists from all over the world. The most recent, the Third Assessment Report: Climate Change 2001 (TAR) and Fourth Assessment Report: Climate Change 2007 (AR4), are going to be mention throughout this chapter to provide the reader with the most accurate information available related on global warming.

2.1.1 Intergovernmental Panel on Climate Change

Global temperature has increased significantly during the 20th century with an average surface temperature between 0.56 to 0.92 °C (Walthall et al., 2012). The Intergovernmental Panel on Climate Change (IPCC) through the Third Assessment Report (TAR) highlights that the highest temperatures occurred during the second half of the 20th century in at least the previous 1,300 years (IPCC, 2007).

The linear warming trend corresponding to the years from 1901 to 2000 given by the TAR was estimated to be 0.6 °C. Five years later, the estimated trend for 1906 to 2005 increased to 0.74 °C. In the last 50 years the trend almost doubled the temperature per decade reaching 0.13 °C / D ranking in the

instrumental record of global surface temperature as the warmest from 1995 to 2006 since 1980 (IPCC, 2007)

These increased temperatures have had also an impact on the sea levels causing to expand seawater up to 3000 m in depths around the globe. From 1993 to 2003 the global average sea level increased 72% from the previous 42 years. Observations since 1961 to 2003 show that the average rate ranked between 1.3 to 2.3 mm per year whereas from 1993 to 2003 the rank was between 2.4 to 3.8 mm per year. These increased rates can be attributed to the melting of mountain glaciers (IPCC, 2007). See figure 2.1

The temperatures in the Arctic have increased by 3 °C since the 1980. Therefore, the area covered with ice has been affected decreasing approximately by 7% and up to 15% during the spring season. Observation done by satellite prove an ice shrinkage of 2.7 % per decade since 1978 as a consequence of the almost twice global average rate increased temperature in the past 100 years in this area (IPCC, 2007).

Although information is limited to some regions, a change in precipitation has been observed around the globe. Rainfall has increased in the eastern parts of South and North America, Northern Europe, central and northern Asia while The Mediterranean, south Africa, Sahel and southern Asia regions have been dryer from 1900 to 2005 (IPCC, 2007).

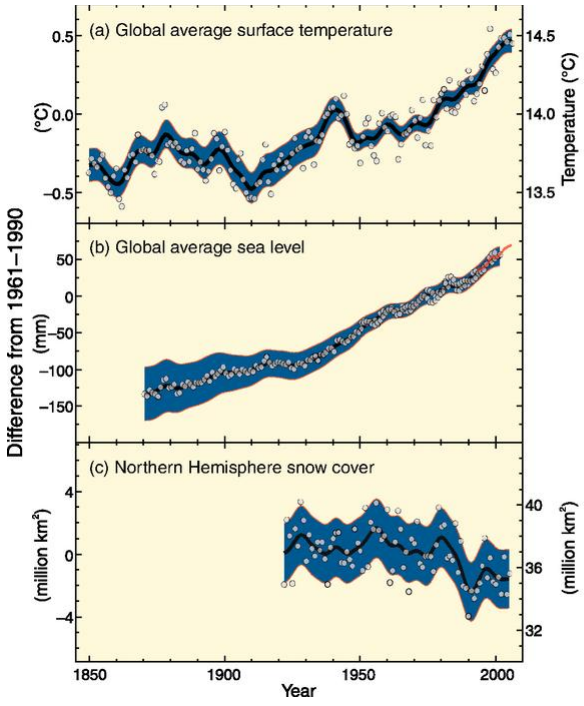


Figure 2.1: Changes in Temperature, Sea Level and Northern Hemisphere Snow Cover (IPCC, 2007)

2.2 Greenhouse Effect

The changes in temperature that have been observed are attributed to the increase amount of anthropogenic greenhouse gases that are being released to the atmosphere. As stated by the TAR "most of the observed warming over the last 50 years is likely to have been due to the increase in greenhouse gas concentrations" (IPCC, 2007).

The greenhouse effect is a natural phenomenon in which heat is trapped between the surface of the earth and the atmosphere. The greenhouse gases include carbon dioxide (CO_2), water vapor (H_2O, g), methane (CH_4) and nitrous oxide (N_2O). As the sun hits the earth, these gases allow solar energy to penetrate the atmosphere but prevent most of this energy from escaping, keeping the earth warm. Most of the energy is absorbed by oceans and land, radiating heat outward from the surface. Greenhouse gases trap some of the radiated heat and re-emit the energy in all directions. The energy kept helps to warm the earth while other energy is released back into space. Without the presence of these gases in the atmosphere or the atmosphere itself the solar energy would be lost. Therefore the temperature of the earth would be below the freezing point. However, as the concentration of these gases increase in the atmosphere, so does the greenhouse effect causing the earth's temperature to rise (National Research Council of the National Academies, 2012). Figure 2.2 shows an amplification of the greenhouse effect.

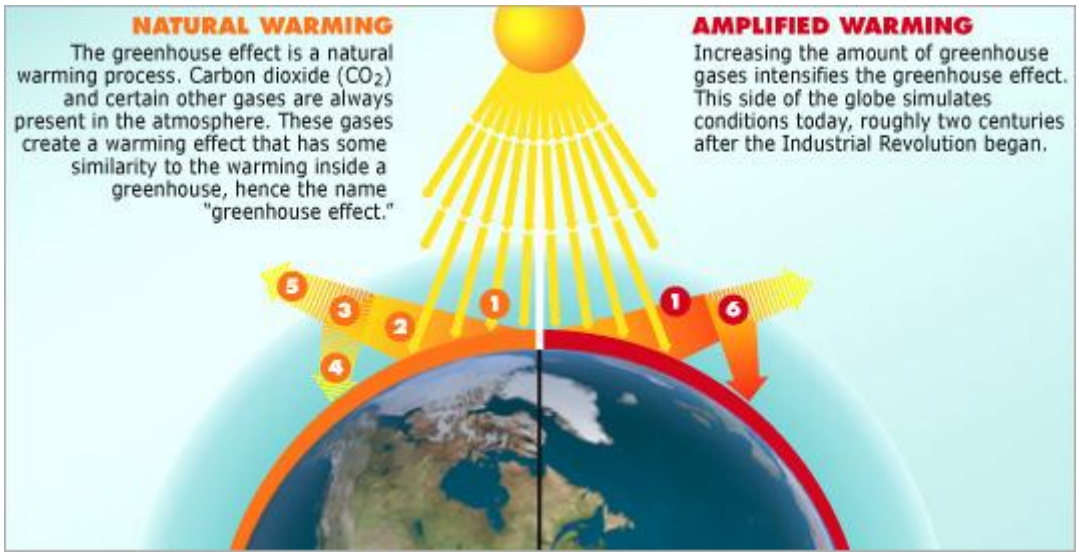


Figure 2.2: Greenhouse Effect

(National Research Council of the National Academies, 2012)

2.2.1 Experimental work

Since around 1850s, many scientists have studied and recorded global warming and the greenhouse effect. Carefully collecting, analyzing data and formulating models to study the climate change process to make assumption. As new data and technology becomes available hypothesis on what is causing global warming and the effect of the greenhouse gases in the atmosphere can be ruled out while others prevail. For more than a century, scientists have agreed that the main factor that affects climate change comes from the level of concentrations of CO_2 and H_2O in the atmosphere, which are known as greenhouse gases (Le Treut et al., 2007).

An experiment performed by John Tyndall in 1859 revealed that the increased molecules such as water or carbon dioxide in the atmosphere can be linked to the change of climate. In his laboratory experiment, Tyndall identified the absorption of thermal radiation by molecules present in H_2O and CO_2. Svante Arrehenius, in 1895, concluded that a 40% of decrease or increase of CO_2 in the atmosphere could cause glacial growth or reduction. His study was based on a greenhouse gases climate prediction. It has been found that the amount of CO_2 varies between interglacial and glacial periods. But, through a century of investigation, it can be concluded that climatic change is the result of changes in CO_2in the atmosphere. In 1938, G. S. Callendar, was able to link climate change and greenhouse gases. Using a set of equations, he found that when the concentration of CO_2 in the atmosphere was doubled the result was an average of 2°C increase in the global temperature, observing higher temperatures at the poles. He also proved that the rise in CO_2in the atmosphere was connected to the increased fossil fuel combustion (Le Treut et al., 2007).

A study in the North Atlantic zone of the Arctic in 1947 by the scientist Ahlmann, revealed an increase of 1.3°C since the past century and. In a later study in 1956, a similar model was used by the scientist Plass to forecast climate change. In the 1950s, CO_2 and H_2O were the only two compound considered as greenhouse gases, which were previously discovered by Tyndall. Three decades later, other gases such as CH4, N2O and CFCs were also found to be significantly important. Furthermore, the reflecting sunlight effect of the aerosol-cloud discovered by Twomey in 1977 and atmospheric aerosols were also beginning considered as climate change constituents (Le Treut et al., 2007).

2.2.2 The Human Fingerprint on Greenhouse Gases

The globe has its own natural carbon cycle in which carbon is exchanged between oceans, biosphere and atmosphere. With this carbon exchange the earth is able to maintain stable average temperatures. However, human activities such as the increasing amount of burning fossil fuels and

deforestation had impacted the amount of carbon dioxide in the atmosphere. These human activities are known as "The human fingerprint on greenhouse gases" (University of New Hampshire).

In order to separate the natural global carbon cycle from those due to the effect of human activities, Chares David Keeling initiated an accurate measurement of CO_2 concentrations in the atmosphere in 1958. The experiment was performed on Mauna Loa in Hawaii and it consisted on documenting a time series of the seasonal exchange of CO_2 between the ocean, biosphere and atmosphere. Later comparisons on the amount of CO_2 in the atmosphere identified the fossil fuel burning as the main contributor of the increased concentration. (Le Treut et al., 2007)

Another analysis was done on air found in ice bubbles from Antarctica and Greenland. The measurements proved that the concentrations of CO_2 were significantly lower throughout the last ice age than in the past 10,000 years. The results of these analysis showed that the concentrations of CO_2 ranged from 280 ± 20 ppm up to the year 1750. To have a better perspective on the magnitude of the CO_2 increment, since the industrial period, these concentrations have rose reaching 367ppm in 1999 and 379 ppm in 2005. (Le Treut et al., 2007)

Other greenhouse gases, such as methane and nitrous oxide, have been detected through direct measurements of the atmosphere since 1970. The first measurements showed that methane was increasing 1% per year but, during the 1990s, decreased to 0.4% per year. Nitrous oxide shows a smaller concentration on the atmosphere with only 0.25 % increase per year which makes it more difficult to detect. In order to compare this analysis over time other measurements were made from air found in snowpack which was estimated to be initially trapped 200 years ago. The results proved an increment in both N_2O and CH_4 over the 20^{th} century. Through this analysis, scientists were able to determine the average abundance in CH_4 back 1 thousand years up to the 19^{th} century, recording 700 ppb. To compare this findings with the present time, in 1998 the concentration of CH_4 reached 1,745 ppb and 1,774 ppb in 2005. The 150% increment in the concentration of CH_4 can be attributed to increasing anthropogenic emissions made in over the present industrial era. For N_2O, a rage between 180 to 260 ppb was identified during the glacial-interglacial cycles and increasing 15% over the industrial period scoring 314 ppb in 1998 and 319 ppb in 2005 (Le Treut et al., 2007).

The amount of synthetic halocarbons in the atmosphere, which are also greenhouse gases with high global warming potential, began to increase until the 1990s. This is because these compounds did not exist before, as scientists have not found any traces through ice core researches corroborating the human fingerprint. Figure 2.3 is a graphical representation of CO_2 concentrations and emissions. The top part (a) the monthly average CO_2 concentrations taken by Keeling and Whorf in Mauna Loa, Hawaii

from 1970 to 2005 is represented in black, New Zealand by Baring Head in blue, Canada's sample measurements of atmospheric oxygen in pink, Australia in 2006 in cyan. The bottom part (b) fossil fuel burning emissions of CO_2 annually and cement manufacture in GtC per year is represented in black, annual averages of the 13C/12C ratio of atmospheric CO_2 measurements taken by Keeling in Hawaii from 1981 to 2002 in red (Le Treut et al., 2007).

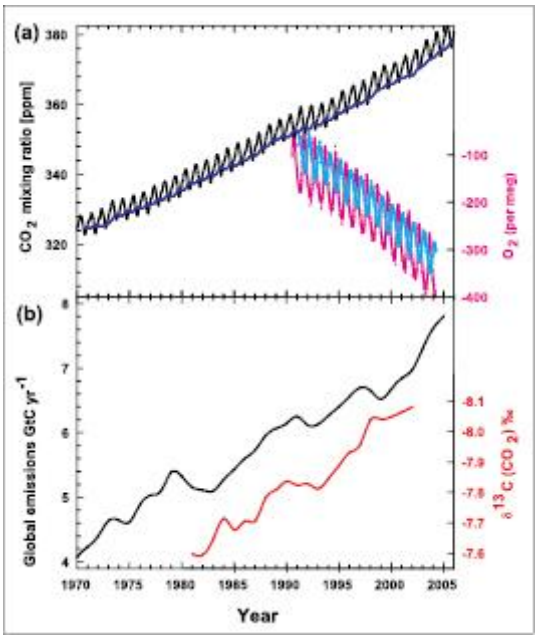


Figure 2.3: CO_2 Concentrations and Emissions (Le Treut et al., 2007)

2.3 Methane

The second most emitted greenhouse gas in the United States is Methane. Although this gas is emitted through natural processes, human activities are also causing emissions raising the concentration levels in the atmosphere. In the US, about 9% of the greenhouse gases emitted in 2011 came from Methane. This percentage only accounts for human activities. Even though methane's lifetime is shorter than carbon dioxide, as it is naturally removed from the atmosphere, it traps more radiation than CO_2. Comparing the effect on climate change over 100 year's period of these two gases, methane's impact is 20 times greater than carbon dioxide. (United States Environmental Protection Agency, 2013)

The emissions of CH_4 that are produced through human activities accounts for 60% of the greenhouse gases worldwide. For instance, the primary component of the natural gas is methane, which is produced by industries and is the largest contributor of CH_4 emissions in the US. The emissions occur during its production process, storage and distribution. Also, methane gas is emitted in the extraction of

petroleum as it is frequently found next to it. Therefore petroleum refinement, transportation and storage cause emissions of CH_4 as well. Also, the storage of animal's manure produces high levels of CH_4 emissions. Animals raised for food produce CH_4 emissions through their digestive process, making agriculture the primary cause of methane emissions in the world. Furthermore, the decomposition of waste from home and businesses produces methane. In the US, the third largest source of methane comes from landfills as seen in figure 2.4 (United States Environmental Protection Agency, 2013).

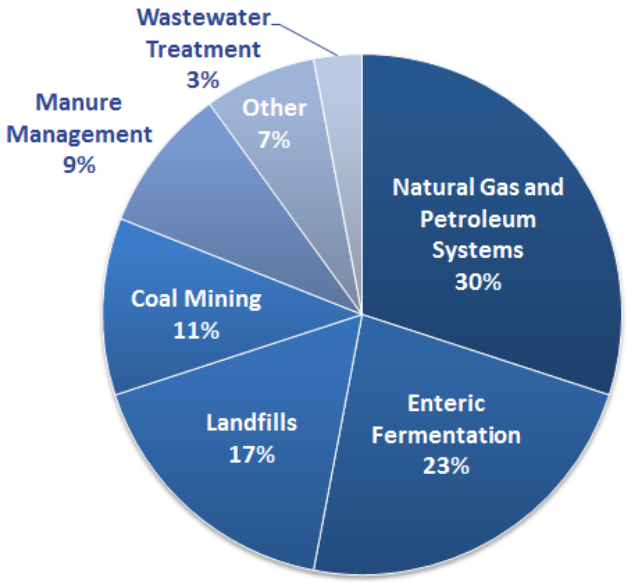


Figure 2.4: Estimated Emissions from the Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2011

2.3.1 Methane Atmospheric Concentration

The Global Monitoring Division of the National Oceanic and Atmospheric Administration (NOAA) is a scientific agency within the United States Department of Commerce that covers the widest geographically network site in the world. This agency analyses atmospheric elements that affect climate change through long-term measurements at specific locations in the world. A study performed by this agency in 2005, in which 40 different surfaces in both hemispheres were tested, determined that the global average concentration of CH_4 was 1,774.62 ± 1.22 ppb. This network assumes a 90% confidence interval, due to the distribution sampling location uncertainty, which is calculated through the Monte Carlo technique (Forster et al., 2007).

The Advanced Global Atmospheric Gases Experiment (AGAGE), sponsored by NASA, has measured and documented the atmospheric composition since 1978. This research agency also monitors

methane in the atmosphere in five locations in the north and south hemispheres. Thirty-six methane samples are taken at each location every day though an automated system. The annual mean level of CH_4 in the atmosphere for 2005 reported by the AGAGE was 1,774.03 ± 1.68 ppb (Forster et al., 2007).

Even though the atmospheric concentrations level has increased in the last 650,000 years by about 30%, measurements made by NOAA and AGAGE at different locations in both hemisphere show that the growth rate decreased about 1% per year between the late 1970's and early 1980's, approaching almost zero by the end of the 1990's. Although there is no real explanation for the decline in annual rate some scientists have related this event to the balancing between sources and sinks. Dulgokencky et al., (1998) estimated that the average source strength between the period of 1984 and 1997 was about 550 Tg per year including a soil sink term of 30 Tg per year with a decreased lifetime of 8.6 years and implies a source strength average of 570 Tg per year. Francey et al., (1999) also concluded that, after 1982, the decline rate was consistent with constant or almost constant CH_4 sources and constant OH. However, other analysis contradicts these theories (Forster et al., 2007)

2.4 Biogas

Biogas is produced by anaerobic digestion through the fermentation of organic material. It is mainly composed of methane with a rage of 50 to 80% in its concentration. Carbon dioxide covers 20 to 50% and small amounts of hydrogen, carbon monoxide and nitrogen. Biogas is a renewable energy source which can be combusted to create power. It can be used to replace natural gas and to generate electricity with a 41% electrical efficiency. On the other hand, its efficiency when used as a fuel cell exceeds 60%. Furthermore, biogas can be used as a vehicle fuel with greater efficiency and reduced carbon dioxide emissions by 95% (Persson et al., 2006). At last, biogas can also be supplied through the natural gas grid. Table 2.1 shows the different technologies that can use biogas to generate energy

Table 2.1: Biogas to generate energy by technologies (Persson et al., 2006)

FEATURE	PETRIK ENGINE SI	DIESEK ENGINE JET INGNITION	DIESEL ENGINE SI	MICRO TURBINE
Efficiency [%]	24-29	30- 38	35- 42	26- 29
Maintenance cost	High	High	Medium	Low
Investment cost	Low	Medium (high)	Medium	High
Power [kW]	5- 30	30- 200	>200	<100
Lifespan	Low	Medium	High	High

2.4.1 Anaerobic Digestion Biochemical Process

The anaerobic digestion consists of three basic processes: hydrolysis, acetogenisis and methanogenisis (Figure 1.5). Through this process, anaerobic bacteria ferment biodegradable materials such as manure, sewage, municipal waste, green waste plant material and crops, in the absence of oxygen to produce biogas. Through hydrolysis, cellulose, proteins and fats, which are insoluble materials, are broken down into soluble compounds. In the next stage, these soluble compounds are transformed into organic acid by acetogenic bacteria. Finally, the organic acids produced are converted by methanogenic bacteria into methane, carbon dioxide and water. Also, small amounts of hydrogen sulfide and ammonia are produced (Bramley et al., 2011). Figure 2.5 shows the stages of the anaerobic digestion process.

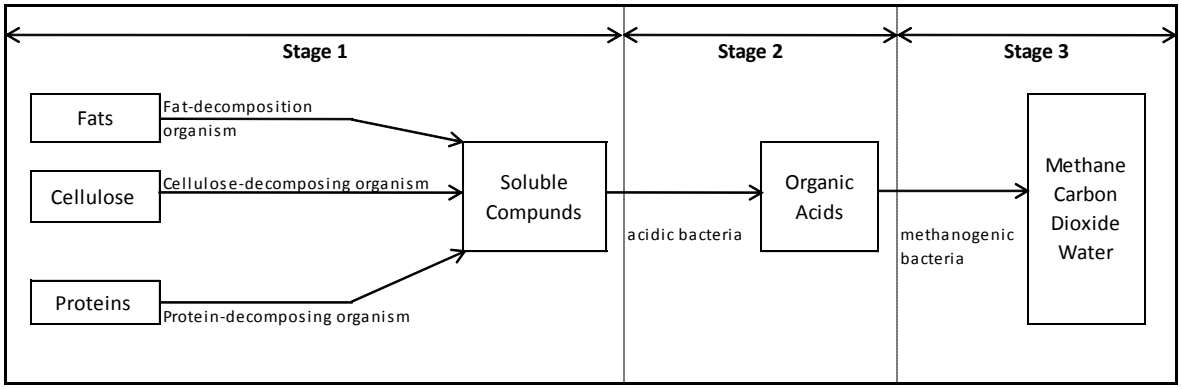


Figure 2.5: Anaerobic Digestion Process

There are many factors that affect the digestion process and the most important is temperature. Changes in temperature slow bacterial activity so a constant temperature is required to optimize the process. The temperature ranges in which anaerobic bacteria work most effective is between 36.7°C and 54.4°C. Other factors that affect the productivity of the process include the digesting material mix, particle size, pH, amount of water and solids, amount of carbon and nitrogen, and time (Bramley et al., 2011)

2.4.2 Biogas Plants on Farms

Five parts embraces a classic farm biogas plant; manure gathering, anaerobic digester, waste matter storage, gas handling, and gas use.

Manure gathering: A typical farm-based biogas plant is comprised of five parts: manure gathering, anaerobic digester, effluent storage, gas handling, and gas use. Dairy, swine and poultry manure can be use to produce biogas, however, some pre-treatment is usually required. The manure state

determines the type of anaerobic digester needed. For example, liquid manure has better performance when processed in a film fixed digesters or covered lagoons. Semisolid manure is suitable in a plug flow digester while slurry and solid manure best perform in a complete mixed digester. Manure that contains more than 13% of solids is not viable for production. (Bramley et al., 2011)

Anaerobic digesters: The most popular farm digesters include complete mix, plug flow, covered lagoon (Figure 2.6) and fixed film. The different characteristics are summarized in table 2.2 (Bramley et al., 2011)

Waste matter storage: The waste matter of the manure processed in the digester has a few biodegradable compounds and no remaining odor. It is considered to be biologically friendly and it can be used as a fertilizer as its nutrients become more available to be absorbed. The solid waste can also be used for dairy cattle bedding, soil amendment, organic fertilizer, potting soil and compost. (Bramley et al., 2011)

Gas handling: Anaerobic digesters require a gas handling system to transport the biogas obtained. The system comprises of gas pumps, gas meters, piping, pressure regulators and occasionally, a gas scrubber to prevent equipment corrosion (Bramley et al., 2011).

Gas use: As previously mention, biogas can substitute natural gas. However, it must be treated to obtain similar properties. The most common application for the biogas obtained on small scale farm's digesters is the direct combustion for cooking heating and lighting. Also, when biogas is pretreated to improve its quality it can be distributed to households through conventional gas pipes. Countries such as Sweden, Germany, Switzerland and France are currently applying this technique. Furthermore, biogas can also be use to generate electrical power or combined heat and power. It also replaces conventional vehicle fuel and these changes are seen in countries such as Germany, Sweden, Spain, Australia, China, India and the United States. However, this application is limited to the availability of gas stations that provide biogas. (Bramley et al., 2011)

Figure 2.6: Digester Designs: Covered Lagoons (left), Complete-mix (center), Plug-Flow (right) (AgSTAR)

Table 2.2: Different characteristics of Anaerobic Digester designs (AgSTAR).

Characteristic	Coverd Lagoon	Complete Mix Digester	Plug Flow Digester	Fixed Film
Digestion Vessel	Deep Lagoon	Round/Square In/ Above-Ground Tank	Rectangular In-Ground Tank	Above Ground Tank
Level of Technology	Low	Medium	Low	Medium
Supplemental Heat	No	Yes	Yes	No
Total Solids	0.5- 3%	3- 10%	11- 13%	3%
Solid Characteristics	Fine	Coarse	Coarse	Very Fine
HRT*(days)	40-60	15+	15+	2-3
Farm Type	Dairy, Hog	Dairy, Hog	Dairy Only	Dairy Hog
Optimum Location	Temperature and Warm Climate	All Climates	All Climates	Temperature and Warm
*Hydraulic Retention Time (HRT) is the average number of days a volume of manure remains in the digester				

2.5 USA

By the end of 2011, there were 176 anaerobic digesters that processed livestock manure in the United States. Since 2000, there has been an average of 16 new digesters each year and the annual electricity generation increased from 14 million kWh to around 331 million kWh per year although the potential to grow is much greater as this represents only 2% of the manure sector. By 2009, 17% of the facilities were producing biogas for reasons other than creating energy such as boiler fuel, injection into natural gas grids and odor control with an increase from less than 1 million kWh to 54 million kWh, from 2000 to 2009. More farms are being able to supply their own energy and control waste storage. As a secondary effect, greenhouse gas emissions are reducing and there is more odor control.

Since 2000, farm digester systems have produced biogas supplying energy in different forms. Most of the biogas produced has been transformed into electricity reaching 385 million kWh in 2009. An increase of 19 million kWh was estimated to be produced the following year, with an approximate total of 404 million kWh. The estimated energy outputs by anaerobic digesters from 2000 to 2009 as well as the estimated production of energy by states are shown in figures 2.7 and 2.8 (United States Environmental Protection Agency, 2010).

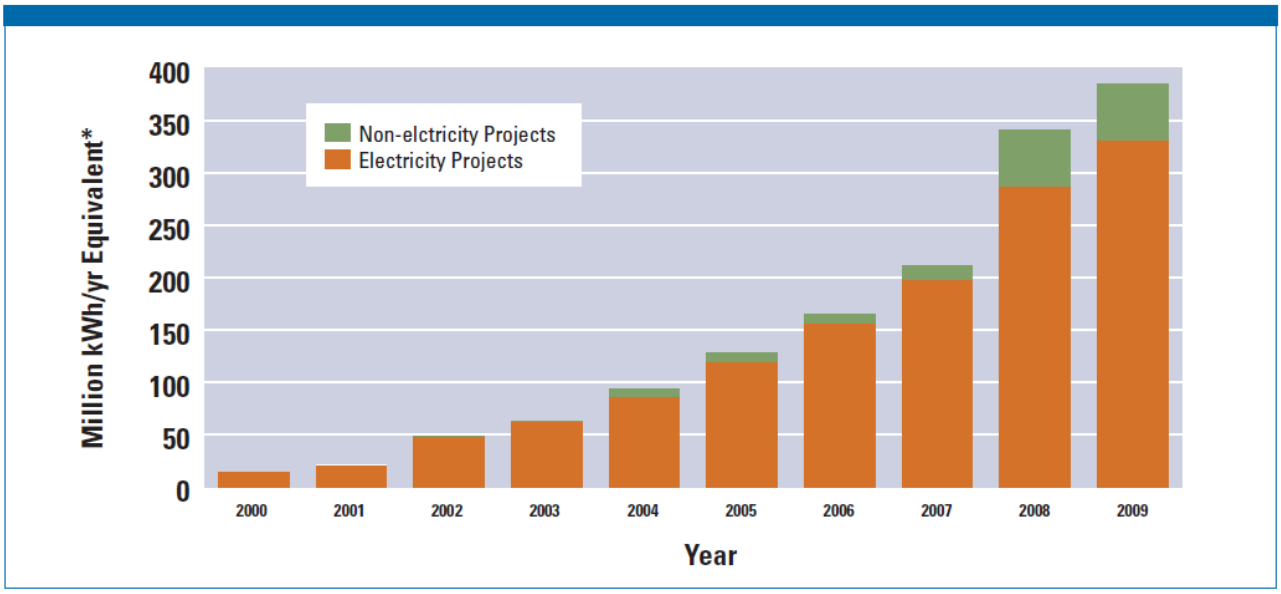


Figure 2.7: Energy produced by anaerobic digesters in the U.S. (EPA)

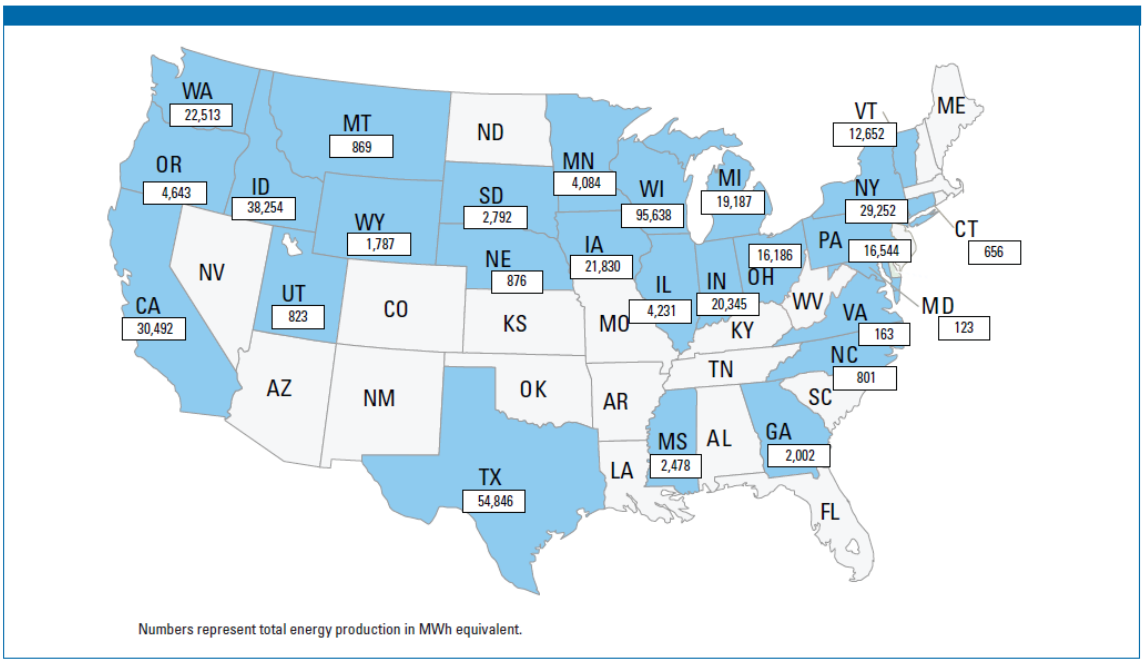


Figure 2.8: Estimated annual protection of farms digesters' energy as of July 2010 (EPA)

In 2011, about 541 million kWh of energy were produced through anaerobic digestion for which 489 million kWh were transformed into electrical power, sufficient to provide one year of electricity to 36,000 U.S. homes.

2.5.1 2011 Trends

About 63% of the new digester that operated in 2011 were composed of complete mix and mixed plug flow designs, as seen in figure 2.9. The rest includes covered lagoons, horizontal plug flow as well as other types of anaerobic systems. While the majority of these technologies use livestock manure and are owed/operated in farms, about 30% operate with organic waste such as food and agricultural wastes. Also, there has been an increase in the number of digesters owned and operated systems by others than those of farms (United States Environmental Protection Agency, 2010).

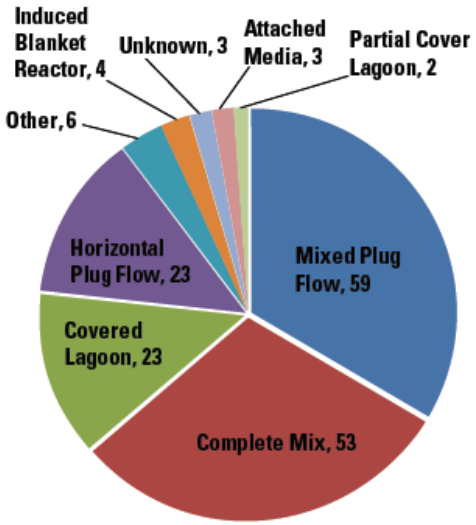


Figure 2.9: Operating systems by technology (EPA)

From 2003 to 2011 the average energy created through anaerobic digesters raised from 125 kW to 454 kW in the U.S. This raised is the result of the increased numbers of small-medium sized digesters operating in farms as well as large farm and centralized systems. Figure 2.10 shows the operating manure digesters by state in 2011. A total of 176 operating projects with an estimated energy production of 541,000 MWh/yr. (United States Environmental Protection Agency, 2010)

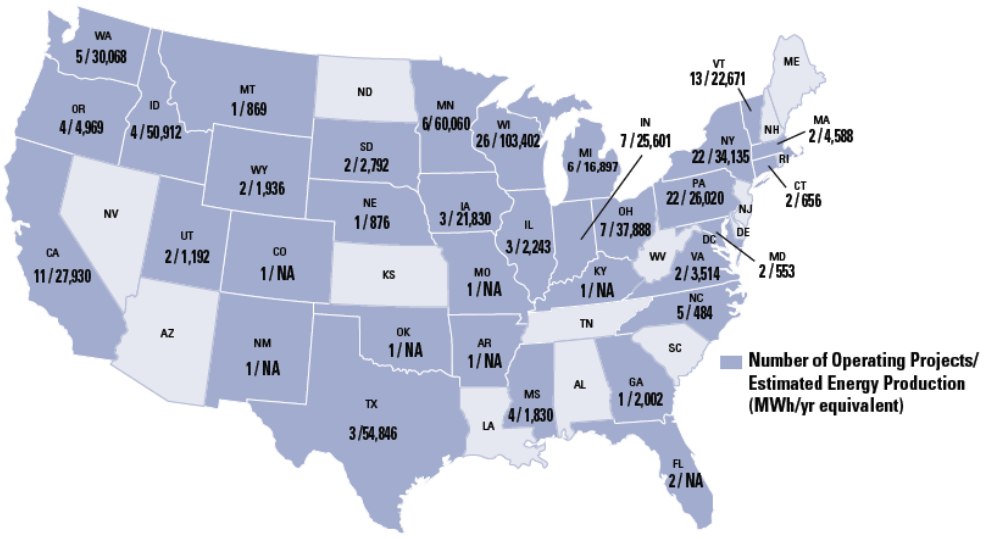


Figure 2.10: Operating manure digesters in 2011 (EPA)

2.5.2 Reduction of Carbon Dioxide and Methane Emissions

The use of anaerobic digesters to create energy reduces more than 1 million metric tons of CO2 emissions. This reduction happens in two ways. Manure that is left to decompose emits CH_4. The digester process helps to capture and destroy this greenhouse gas avoiding its natural emissions to the atmosphere. Also, the energy produced by digesters substitutes the energy created from fossil fuels.

In the U.S. 55,000 metric tons of methane emissions were reduced and 301 metric tons of carbon dioxide emissions were avoided through the use of anaerobic digesters in 2011. To put this in a different perspective, this is the same as taking away around 294,000 cars, reduce the consumption of gasoline by more than 168 million gallons or reduce the consumption of oil by nearly 3.5 million barrels (United States Environmental Protection Agency, 2010)

Figure 2.11 illustrates an estimate of the avoided emissions from 2001 to 2009. The data was calculated based on EPA eGRID national average emission rates for electricity projects and EPA's *Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2006* for non-electricity projects.

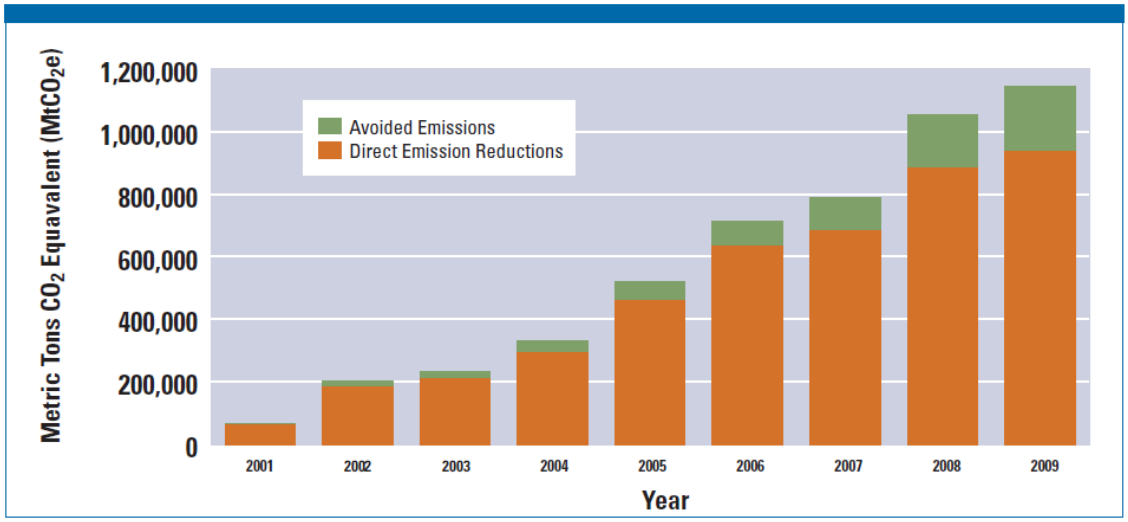


Figure 2.11: Estimated emission reductions from anaerobic digesters in the U.S (EPA)

Conclusions

In this chapter, information on the third and fourth assessment reports published by the Intergovernmental Panel on Climate Change was presented to demonstrate the increasing temperatures the globe has experienced in the last century. The most resent, The Fifth Assessment Report (AR5) released on September 27th of the present year, was not included in the chapter as it was released after this chapter was written. The information provided here shows that emissions of CH_4, which is the second most emitted gas in the US, contributes for 60% of the greenhouse gases worldwide produced through human activity. These emissions can be reduced by creating renewable energy in the form of biogas that is produced through anaerobic digestion. In the next chapter, some optimization algorithms that can be applied to the scheduling of feedstocks in anaerobic digesters are going to be presented

Chapter 3: Optimization Algorithms

Chapter 3 will review a few meta-heuristic methods that were contemplated to solve the scheduling problem of a single anaerobic digestion considering multiple feedstocks. The general descriptions of the methods as well as their applications are presented.

3.1 Ant Colony Optimization

The ant colony optimization is a meta-heuristic algorithm developed by Marco Dorigo from the Electric Department in the Politecnico of Milano in the early 1990's as an intelligent approach to optimize complex combinatorial problems such as the travel salesman (Colorni et al., 1991). Since its development, it has been apply to solve optimization problems in different fields. For instance, to optimize water distribution systems (Zecchin et al., 2006), to minimize the total completion time in a shop scheduling problem (Shyu et al., 2004), to solve the vehicle routing problem with time windows (Ding et al., 2012), to design a neuro-fuzzy controller for real-time control of an inverted pendulum (Baojiang and Shiyong, 2007) and to develop partially constraint ACO algorithms for the solution of optimization problems with explicit constraints (Afshar, 2007) among others.

An interesting fact about ants is that, in spite of their visibility limitations, these little insects can set up a network with the shortest path from their colony to feeding source. Ants communicate between one another by leaving a pheromone trail marking the path followed. Therefore, the more ants following the same path the stronger the pheromone scent becomes. As isolated ants moving randomly come across with different trails can decide to follow one with a higher possibility of choosing the strongest scent path reinforcing it with its own pheromone. This group activity is a form of autocatalytic behavior and the objective of the ant colony optimization is to mimic this behavior (Colorni et al., 1991). The example in figure 3.1 illustrates how ants identify the shortest path around an obstacle.

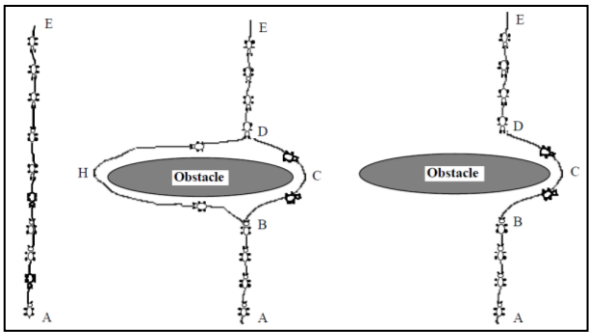


Figure 3.1: Shortest Path Identification around an Obstacle (Colorni et al., 1991).

In the first path ants are following the same trail from point A to source E and vice versa. After an obstacle is suddenly introduced, the ants separate leaving two different trails of pheromones with same probability of preceding ants to take. Those ants who chose to take path BCD will reach point D before those who took path BHD. The result is that the ants coming from the opposite direction (E to A) will find a stronger scent at point D and will find more attractive to take path DCB. As more ants take the shortest path, the pheromone scent gets stronger and the probability of other ants to take this path increases. Path BHD becomes weaker until the pheromone scent disappears resulting in all the ants following the shortest path (Colorni et al., 1991).

The pheromone values are updated by all the ants that have completed the tour with the following equation

$$T_{ij} \leftarrow (1-\rho) * T_{ij} + \sum_{m=1}^{M} \Delta T_{ij}^{m}$$

Where:

$\rho =$ Evaporation rate

$M =$ Number of ants

$\Delta T_{ij}^{m} =$ The pheromone quantity laid on edge (i, j) by the *m*th ant

$$\Delta T_{ij}^{m} = \begin{cases} Q/L_m & in\ ant\ k\ travels\ on\ edge\ i,j \\ 0 & otherwise \end{cases} .$$

Where:

$Q =$ A constant

$L_m =$ The tour length of the *m*th ant.

In the solution process, the ants select the next city to visit through a stochastic mechanism. The probability of going to city j, when ant m is in city i and has built the partial solution, is given by

$$P_{ij}^{m} = \begin{cases} \dfrac{[T_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{C_{il} \in N(sP)}[T_{ij}]^{\alpha}[\eta_{ij}]^{\beta}} & if C_{il} \in N(s^P), \\ 0 \ otherwise, \end{cases}$$

Where:

$N(s^P) =$ The set of feasible components; edges (i,l) where l is a city unvisited by ant m

α & β = Parameters to control the relative importance of the pheromones versus the heuristic information η_{ij}, which is given by

$$\eta_{ij} = \frac{1}{d_{ij}},$$

Where:

d_{ij} = Distance between cities i and j (Wu et al., 2009).

3.2 Bee Algorithm

The BA is a new population based algorithm first developed by Pham D.T. and colleagues in 2005. The Meta heuristic algorithm simulates natural behavior of honey bees to find the optimal solution when looking for food. The basic version executes a type of neighborhood search together with a random search that is applied to solve real optimization problems. For instance, to solve complex transportation problems (Teodorovic and Dell'orco, 2005), to route mobile adhoc networks (Chaundhary, 2002), to prioritize the regression test suite based on maximum fault coverage (Kaur and Goyal, 2011), to solve Sudoku puzzles (Pacurib et al., 2009), to solve web searching, function optimization and hierarchical optimization problems (Navrat et al., 2009), for solving and optimize engineering problem (Yang, 2005), to identify accidents in nuclear power plants (Oliveira et. al 2009), to solve the Travelling Salesman Problem (Wong et al., 2010), to solve Multi-Dimensional Knapsack Problem (Nhicolaievna and Thanh , 2008), to solve numerical optimization problem (Lu and Tapkan, 2004) and job shop scheduling optimization (Stanarevic et al., 2011) among others.

In nature, honey bees can travel distances for more than 10 km in different directions simultaneously to explore great quantity food sources. Basically, the flowers that have greater amounts of pollen or nectar must be visited by more bees, while the flowers with fewer amounts obtain less visitations as the recollection effort of these nutrients increases. The search process starts when a group of bees in a colony is sent to look for areas of potential flowers keeping a percentage of its population waiting for the scout bees to return. When these bees come back to deposit the pollen or nectar collected they perform a dance known as the "waggle dance" that is used to measure the sugar content found. This dance is how the colony communicates and it includes three pieces of information about the flowers explored; the direction, distance and quality rating. With this information, bees waiting inside the hide can fly to a specific area without guides (Pham et al., 2006)

The Australian ecologist Karl von Frisch translated the meaning of the "waggle dance" in which the duration of the waggle lasts about 75 milliseconds for every 100 meters the flowers are apart from the hive (Figure 3.2)

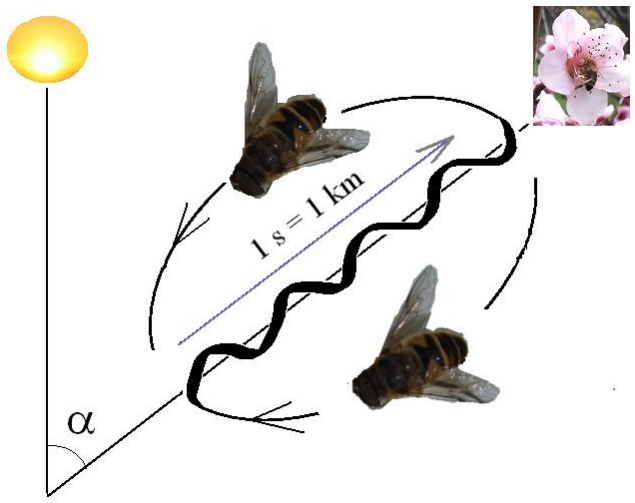


Figure 3.2: Wiggle Dance

The parameters required for the bee algorithm are as follows

n = Number of scout bees

m= Number of sites selected for n to visit

e= Number of best sites from m sites

npe= Number of bees recruiter for best e sites

nsp= Number of bees recruited for the other m-e sites

ngh= Initial size of patches (including sites in its neighborhood & stopping criteria) (Pham et al., 2006)

The bees algorithm include elite bees and best fitted bees selected from a population of size NS. The term Rec refers to the number of recruited bees and it varies linearly from nep to nsp depending to the rank of the bee chosen. The number of remaining bees (NR) uses on third of the population N while the selected bees (NS) uses two thirds. Consequently, the sum of NS and NR equals to the total population size N. The selected bees and recruited bees are in charge of exploring potential flowers whereas the remaining bees move randomly looking for new exploration sites.

To update the locations with better fitness, the selected bees use the equation:

$$cx = x + ngh \times (2r_1 - 1)$$

Where:

x =D-dimensional vector location of the selected bee

cx = Candidate location found by a recruited bee

ngh = Fixed value for neighborhood search

r_1 = D-dimensional random vector with a range of [0, 1]

If $cx > x$, then x is replaced. If not, the next recruited bee is evaluated.

The factor value ngh is difficult to identify because this is related to the search space and the current convergence. Different settings, either with a fixed or a linearly decreasing among iterations value, ranged within [0.008, 20]. A stochastic self-adaptive ngh (ssngh) frees the recruited bees from the ngh setting which follows the formula:

$$ssngh = \frac{\sum_{i=1}^{D} \left| x_{s1,i} - x_{s2,i} \right|}{D} \times r_2$$

Figure 3.3 describes the pseudo code used for the basic bee's algorithm

1. Initialize population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
   //Forming new population.
4. Select sites for neighborhood search.
5. Recruit bees for selected sites (morebees for best e sites) and evaluate fitness.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitness.
8. End while

Figure 3.3: Basic Bee Algorithm – Pseudo Code

3.3 Firefly Algorithm

The firefly algorithm is a relatively new optimization method which was developed by Yang, in 2008, based on the flashing characteristics behavior of fireflies (Yang, 2010). Although this optimization technique is recent it has been applied in different areas to solve complex problems. For instance, to solve non-convex economic dispatch problems with valve loading effect (Yang et al., 2012), to solve discrete optimization problems (Sayadi et al., 2013), to optimize the operation of a Micro-Grid based on an efficient Point Estimate Method (Mohammadi et al., 2013), for auto-tuning mobile networks with the aim of achieving energy savings in access networks (Bojic et al., 2012) and to implement it in the loading pattern optimization of nuclear reactor core (Poursalehi et al., 2003).

The flashing characteristic behavior of fireflies includes three main rules:

- Fireflies are attracted to one another in spite of their sex, as they all share the same gender (unisex)

- Attractiveness is proportional to their brightness. That is, given two flashing fireflies, the one with less brightness follows the brighter one. The attractiveness decrease as the distance between them increases. When there is no brighter firefly, then they move randomly.
- Brightness is determined by the objective function (Yang, 2010).

Based on these rules a summary is provide in figure 3.4 as a pseudo code for the Firefly Algorithm

```
Objective function f(x),      x = (x_1, ..., x_d)^T
Generate initial population of fireflies x_i (i = 1, 2, ..., n)
Light intensity I_i at x_i is determined by f(x_i)
Define light absorption coefficient γ
while (t < MaxGeneration)
for i = 1 : n all n fireflies
   for j = 1 : i all n fireflies
      if (I_j > I_i), Move firefly i towards j in d-dimension; end if
      Attractiveness varies with distance r via exp[-γr]
      Evaluate new solutions and update light intensity
   end for j
end for i
 rank the fireflies and find the current best
end while
 Postprocess results and visualization
```

Figure 3.4: Pseudo Code for FA

When optimizing a simple maximization problem, the brightness I at a specific location x of a firefly can be chosen as I(x) / f(x). However, attractiveness will vary with the distance r_{ij} between fireflies *i* and *j*. Also, attractiveness varies upon the absorption of light since light intensity decreases with distance. To calculate the variation of light intensity the inverse square law is applied as shown below (Yang, 2009):

$I(r) = \frac{I_s}{r2}$

Where:

$I(r)$ = Light intensity

I_s = The intensity at the source.

For a fixed light absorption coefficient, the distance r will make light intensity to vary.

$I = I_0 e^{-\gamma r}$

Where:

I_0 = The original light intensity.

The combination of both functions can be approximated, when $r = 0$, using the Gaussian equation. The purpose of this is to avoid singularity.

$I(r) = I_0 e^{-\gamma r^2}$

When slower rate of monotonically is needed the following approximation is used

$I(r) = \frac{I_0}{1+\gamma r^2}$

To define the attractiveness, which is proportional to the light intensity observed by adjacent fireflies, the following form is used

$\beta(r) = \beta_0 e^{-\gamma r^2}$

Where:

β_0 = the attractiveness at r = 0

A general monotonically decreasing form can be used as an attractiveness function in the implementation procedure, such as:

$\beta(r) = \beta_0 e^{-\gamma r^m}, \quad (m \geq 1)$

The distance between fireflies i and j is the Cartesian distance

$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2},$

Where:

$x_{i,k}$ = The kth component of the spatial coordinate x_i of the ith firefly.

And the movement of firefly i attracted by firefly j is given by

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - \frac{1}{2})$$

Where $\beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i)$ is due to the attraction and $\alpha(rand - \frac{1}{2})$ is a random term with a random parameter of α uniformly distributed in [0, 1] (Yang, 2009).

3.4 Genetic Algorithm

The Genetic Algorithm (GA) is a search heuristic based on the Darwinian theory of evolution. It was first proposed by John Holland in 1975 and later developed by Goldberg in 1989 and is generally used to solve optimization problems (McCall, 2005). It has been applied in complex problems in many different fields. For instance, the designing a sliding mode control system (Morin et al., 1995), robot trajectory planning (Tian and Collins, 2004), adapting IIR filters (Nambiar and Mars, 1995), low cost design of IIR digital filters (Wilson and Macleod, 1993), design of robust control systems (Goh et al.,

1996), tracking changing environments (Cobb and Grefenstette, 1993), solving the k-partition problem on hyper cubes (Chohoon et al., 1991), job shop scheduling, rescheduling and open shop scheduling problems (Fang et al., 1993), simultaneous design of membership functions and rule sets for fuzzy controllers (Homaifar and McCormick, 1995), pump scheduling for water supply (Mackle et al., 1995) among many others.

There are five components that distinct the GA; chromosome encoding, fitness function, selection, recombination and the evolution scheme (McCall, 2005).

1. Chromosome encoding

A chromosome represents an individual DNA from a population set and it symbolizes a possible solution for the problem being solved (McCall, 2005).

2. Fitness function

This function evaluates each chromosome from the population set in terms of the objective function of the problem and each output represents the solution's quality (McCall, 2005).

3. Selection

The selection process uses fitness to select chromosomes to enter the evolution process. There are many different selection methods used. The Roulette Wheel is the most traditional method. In this technique, a probability of being selected is assigned to each chromosome which is proportional to the fitness values of all chromosomes in the population. Another technique, such as the Random Stochastic Selection, uses the probability of being selected in the fitness proportional method and chooses each chromosome the same number of times. On the other hand, the Tournament Selection takes two random chromosomes and chooses the one with better solution fit. Furthermore, the Truncation Selection eliminates a fixed number of chromosomes with worst fit and then selects at random from the population (McCall, 2005).

4. Recombination

The purpose of recombination is to mix the genetic material of the chromosomes selected simulating the evolution process when organisms reproduce. There are two nondeterministic operators that can be used to recombine chromosomes; crossover and mutation.

Crossover mixes two selected parents, using a random number from 0 to 1 with uniform probability and compares it to a pre determined crossover rate, to produce one or two children. If the random number exceeds the crossover rate then the crossover is not applied and both parents pass to the next generation unchanged. On the other hand, if the random number is less than or equal to the crossover rate, the crossover is applied. The one point crossover operator is the most commonly used

where a point between 0 and n is selected with uniform distribution. The child chromosome will contain the characteristics of the first parent before the crossover point and the characteristics of the second parent after the crossover point as illustrated in figure 3.5. Other crossover alternatives include the 2 or multi point crossover point and uniform crossover (McCall, 2005).

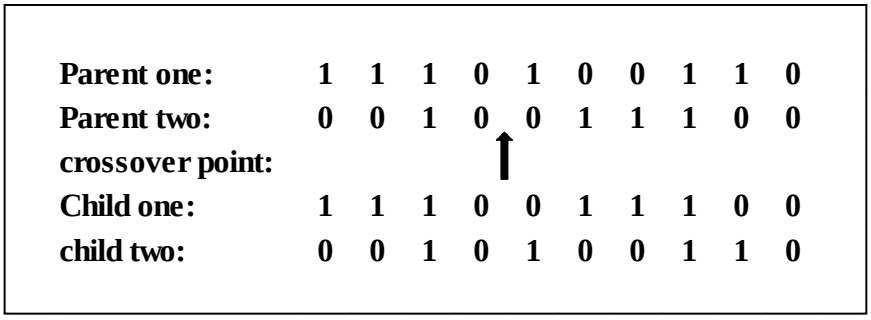


Figure 3.5: One Point Crossover

Once the children are produced, the mutation operator flips one or more allele values to each individual chromosome. The mutation is applied to each position in the chromosome where there are bit-string chromosomes; otherwise, a random number from 0 to 1 with a uniform probability is selected and compared to the predetermined mutation rate which is typically very small. Again, if the random number exceeds the mutation rate then the mutation is not applied at that position. On the other hand, if the number is less than or equal to the mutation rate, then the allele values flip (McCall, 2005).

5. Evolution

In the evolution state, the children chromosomes are passed into the next generation which will become the new set of population. There are several evolutionary techniques that can be used to generate the new populations such as:

- *Complete replacement:* All of the individuals of the descendant generation are created trough selection and recombination.
- *Steady state:* The descendant generation is created by producing only one new chromosome which will replace the less-fit individual from the population source.
- *Replacement with elitism:* Almost all individuals are replaced and a percentage of the best individuals from the source population are kept to prevent the highest possible solutions from being lost in the next generation (McCall, 2005).

A general structure of the Genetic Algorithm is described in figure 3.6.

begin
Let g = 0 be the generation counter
Create and initialize a population, **Pop**(0)
repeat
Evaluate the fitness, $f(x_i(g))$, of each individual x_i of **Pop**(g)
Select individuals from **Pop**(g)
Apply crossover with probability p_c to produce offspring
Apply mutation with probability p_m on offspring
Set population of new generation g = g + 1
until stopping condition is true
end

Figure 3.6: General Structure of the Genetic Algorithm (Galvez et al., 2012).

3.5 Harmony Search

Harmony search is a relatively new music based meta-heuristic algorithm first developed by Zoo Woo Geen in 2001. Since its development, it has been applied to solve optimization problems (Mahdavi et al., 2007), in continuous engineering optimization (Lee, 2005), to obtain optimum core loading pattern for power nuclear reactors (Aghaie et al., 2013), to design water distribution networks (Geem, 2006) vehicle routing problems (Geem, 2005 and Geem et al., 2005) and structural design (Lee et al., 2005) among others.

The Harmony Search was developed with the fundaments of nature-modeled Meta-heuristic algorithms. It is based on the performance process in which a musician improves the state of harmony. For instance, during jazz improvisation, the objective is to find a perfect state through musically pleasing harmony similar to the optimization process in which an optimal solution is being searched to satisfy an objective function. The aesthetic quality is determined by the pitches of the instruments, just like the decision variable values determine the output of the objective function in an optimization problem (Geem and Lee 2004).

To explain how the Harmony Search works, we need to understand the improvisation process of a musician. When improvising, a musician has three options: 1) play a song from memory; 2) play a piece similar to a known song; or create a new song. These options are translated in the Harmony algorithm as: Harmony memory, pitch adjustment and randomization (Yang, 2009).

The harmony memory is very similar to those best-fit individuals used in GA's where only the ones with the better output will be carried over the next generation. In the harmony memory case, the

best harmonies will be kept in the New Harmony memory. A parameter, $r_{accept} \in [0, 1]$, is assigned to ensure the effectiveness of the memory. This parameter is called "harmony memory accepting or considering rate". The lower the parameter is the fewer best harmonies will be selected. In the contrary, when the parameter is higher, most of the harmonies will remain in the memory consequently the risk of potential wrong solution increases since other harmonies are left unexplored. For this reason, the most common rate used is *raccept* = 0.7 ~ 0.95 (Yang, 2009).

The pitch adjustment is determined by a pitch bandwidth b_{range} and pitch adjusting rate r_{pa}. The pitch is linearly adjusted using the following equation

$$x_{new} = x_{old} + b_{range} \times \varepsilon$$

Where:

x_{old} = Existing solution from harmony memory

x_{new} = New solution obtained through pitch adjustment

b_{range} = Pitch bandwidth

ε = Random number between -1 and 1

This procedure generates a new solution around the solution in memory, using a small pitch variation (Yang, 2009).

The pitch adjusting rate, r_{pa}, is applied to control the level of adjustment. Just like the mutation operator in GA's, the lower the rate and narrower bandwidth the slower the convergence of the search as the exploration space becomes only a small subspace of the entire search space. On the contrary, a higher rate and wider bandwidth can cause the answer to spread around possible optimal solutions similar to a random search. Therefore, the most used rate is $r_{pa} = 0.1 \sim 0.5$ (Yang, 2009).

Another factor that helps increase the variety of solutions in the Harmony Search is the randomization. Even when adjusting pitch shares a similar function, it is limited to a local search while the randomization explores different solutions allowing the system to achieve global optimality (Yang, 2009).

The mechanism in which the Harmony Search works is summarized in the pseudo code shown in figure 3.7.

begin
Objective function f(x), $x = (x_1, x_2, \ldots, x_d)^T$
Generate initial harmonics (real number arrays)
Define pitch adjusting rate (r_{pa}), pitch limits and bandwidth
Define harmony memory accepting rate (r_{accept})
while (t < Max number of iterations)
Generate new harmonics by accepting best harmonics
Adjust pitch to get new harmonics (solutions)
if (rand > r_{accept}), choose an existing harmonic randomly
else if (rand > r_{pa}), adjust the pitch randomly within limits
else generate new harmonics via randomization
end if
Accept the new harmonics (solutions) if better
end while
Find the current best solutions
end

Figure 3.7: Harmony Search Pseudo Code

The following steps are a summary of the Harmonic Search.

1. In the first step of the Harmony Search the optimization function F(x) is defined as a minimizing or maximizing problem given an interval $LB_I \leq x_i \leq UB_i$ of decision variables where each x_i is a possible solution. Also, the parameters are defined as: harmony memory size (HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR) and number of imporvisions (NI)
2. In the second step, the harmony memory begins, created from a uniform distribution between the upper and lower bounds. This is performed by $x_i^j = LB_i + r \times (UB_i - LB_i)$, for j = 1, 2, …HMS and r ~ U(0,1).
3. In the third step, a new harmony is improvised generating a new vector $x' = (x_1', x_2', \ldots, x_N')$ considering the following rules; memory, pith adjustment and random sets. The procedure can be explained as:

for every i ϵ [1, N] **do**

if U(0,1) ≤ Harmony Memory Consideration Rate **then** / 'memory consideration'

begin

$x_i^{'} = x_i^j$, where j ~ U(1, 2, … Harmony Memory Size).

if U(0, 1) ≤ Pitch Adjustment Rate **then** / 'pitch adjustment'

begin

$x_i^{'} = x_i^{'} \pm r \times bw$, where r ~ U(0,1) and bw is a random distance bandwidth

end if

else/ 'randome selection'

$x_i^j = LB_i + r \times (UB_i - LB_i)$

end if

end

4. In the fourth step the previous harmony vector generated $x' = (x_1', x_2', \ldots, x_N')$ will replace the worst vector in the harmony memory measured by the objective function.
5. This procedure repeats until the maximum number of improvisations is evaluated. Then stop (Omran et al., 2008).

3.6 Particle Swarm Optimizer

The particle Swarm Optimization (PSO) is a relatively new optimization method that can solve complex optimization problems. It was created by Kennedy and Eberhant in 1995 and it is inspired by the animal's social behavior, in specific fish schooling and birds flocking (He et al., 2004).

This technique has been applied in many different fields to optimize complex problems. Such applications include: The optimization of profiled corrugated horn antennas (Robinson et al., 2002), The analysis for the diagnosis of Parkinson's disease (Eberhart and Hu, 1999), WDM telecommunication networks (Teo et al., 2004), Clustering in large spatial databases (Yan and Ma, 2006), Travelling-sales man problems (Zhi et al., 2004), Traffic flow control (Lu et al., 2006), Power systems (Abido, 2002), Voltage regulation (Olamaie and Niknam, 2006), Generic electromagnetic designs (Grimaccia et al., 2007), Optimization of internal combustion engines (Ratnaweera et al., 2003), among others.

The PSO algorithm include a number of individuals called particles which represent a specific location in a given search space. Each individual is a possible solution to the problem being solved. The individuals move through a multidimensional space to search for higher fitness positions. The initial population of individuals, the particles' velocity and their position are initialized randomly. Throughout the optimization process, each individual memorizes its *local best,* which is the best position found, while the entire population memorizes the *global best*, which is the best position among all individuals. A linearly decreasing inertia weight is used to balance the global and local search capabilities of the individuals. Some advantages of the PSO over other optimization techniques include: simple concept,

easy to execute and computationally efficient. Figure 3.8 illustrates a flowchart of the PSO algorithm (Mohandes, 2012).

The PSO algorithm includes the following steps:

1. Initialization
2. Update Velocity
3. Update position
4. Update best
5. Stopping criteria

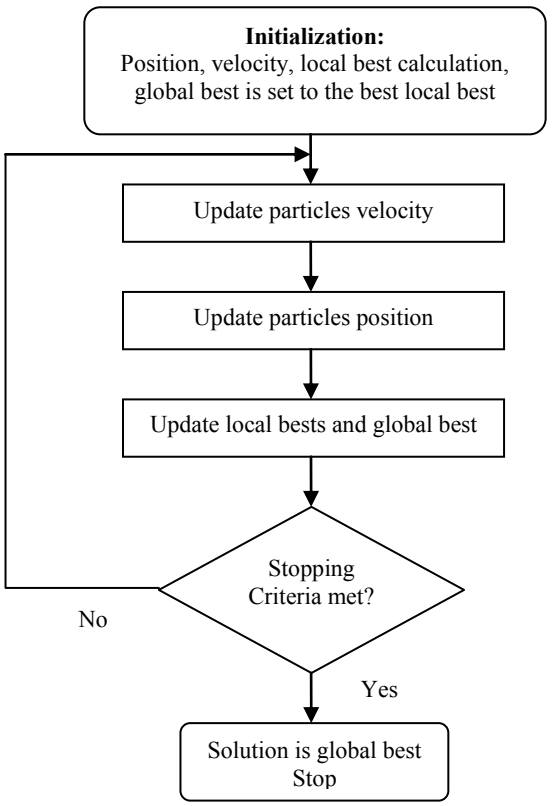


Figure 3.8: Flowchart for the PSO Algorithm

The standard model include two main factors

1. P_i .- Autobiographical memory, (the best previous position of each member in the swarm).
2. P_g.- Publicized knowledge, (best solution found in the current population).

Step 1. In the initial stage, n position and velocity vectors are generated randomly

1) Current position in an N-dimensional search space:

$$X_i^k = (x_1^k, \dots, x_n^k, \dots, x_N^k)$$

Where:

$x_n^k \in [l_n, u_n]$;

$1 \leq n \leq N$

l_n= Lower bound of the nth search space

u_n= Upper bound of the nth search space

2) Current velocity

$$V_i^k = (v_1^k, \dots, v_n^k, \dots, v_N^k)$$

Bounded by:

Maximum velocity $V_{max}^k = \left(v_{max,1}^k, \dots, v_{max,n}^k, \dots, v_{max,N}^k\right)$

Minimum velocity $V_{min}^k = \left(v_{min,1}^k, \dots, v_{min,n}^k, \dots, v_{min,N}^k\right)$.

Steps 2 & 3. The populations' velocity and position are updated with the following equations:

$$v_i^{k+1} = \omega V_i^k + c_1 r_1\left(P_i^k - X_i^k\right) + c_2 r_2\left(P_g^k - X_i^k\right)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}$$

Where:

P_i= Best previous position of the ith particle

P_g= Best position among all particles in the swarm

Which are given by:

$$P_i = \begin{cases} P_i : f(X_i) \geq P_i \\ X_i : f(X_i) < P_i \end{cases}$$

$$P_g \in \{P_0, P_1, \dots, P_m\} \big| f(P_g) = \min(f(P_0), f(P_1), \dots, f(P_m))$$

Where:

f= Objective function

$m \leq M$

M= Total number of particles

$r_1 \sim U(0,1)$

$r_2 \sim U(0,1)$

ω = Inertia weight

c_1, c_2 = Acceleration constants

The combinatorial Particle Swamp Optimization (CPSO) is an important factor in the construction approach which is updated by the following equation

$$V_i^{k+1} = \chi\left(V_i^k + c_1 r_1\left(P_i^k - X_i^k\right) + c_2 r_2\left(P_g^k - X_i^k\right)\right)$$

Where:

χ = Construction factor, given by:

$$\chi = \frac{2}{\left|2 - \varphi + \sqrt{\varphi^2 - 4\varphi}\right|}$$

Where:

$\varphi = c_1 + c_2, \varphi >$

Step 4. Each individual is evaluated according to the updated position to measure its fitness. If the position gives a betters result the local best and global best are updated.

Step 5. This process is repeated until a specified number of iterations are reached or the specified objective function value is reached.

The pseudo code belonging to the Particle Swarm Optimizer in shown in figure 3.9.

```
Loop
    For i = 1 to number of individuals
        if G(x_i) > G(p_i) then do                    //G( ) evaluate fitness
            For d = 1 to dimensions
                p_id = x_id                           //p_id is best so far
            Next d
        End do

        g = i                                         //arbitrary
        for j = indexes of neighbors
            if G(p_j) > G(p_g) then g = j             //g is index of best performer
                                                      in the neighborhood

        Next j
        For d = 1 to number of dimensions
            v_id(t) = v_id(t − 1) + φ_1(p_id − x_id(t − 1)) + φ_2(p_gd − x_id(t − 1)))
            v_id ∈ (−V_max, + V_max)
            x_id(t) = x_id(t − 1) + v_id(t)
        Next d
    Next i
Until criterion
```

Figure 3.9: PSO Pseudo Code

Conclusion

In this chapter, general descriptions of the methods that can be applied to solve complex problems such as scheduling problems were presented. In the next chapter, the description of the scheduling problem of a single anaerobic digester with multiple feedstocks is going to be presented. Also, the methodology to solve this problem is going to be described and some case studies are going to be solved.

Chapter 4: Feedstock Scheduling in Anaerobic Digesters

Many environmental benefits derived from the production and utilization of biogas from anaerobic digestion. Its production helps avoid greenhouse gas emissions such as methane and nitrous oxide, which contribute to climate change, which would otherwise be directly released to the atmosphere by natural decomposition of biomass. Moreover, it reduces carbon dioxide emissions by offsetting conventional fossil fuels such as lignite, coal, oil and natural gas.

According to the Environmental Protection Agency in 2011, around 541 million kilowatt-hours (kWh) of usable energy were produced in the U. S. by digester systems. Using the U.S. EPA's 2011 LFGE Benefits Calculator, this amount of energy can supply over 36,000 average U. S. homes for a year. Furthermore, 55,000 metric tons of methane emissions were reduced and 301,000 metric tons of carbon dioxide emissions were avoided by offsetting fossil fuels. The U. S. EPA's Greenhouse Gas Equivalencies Calculator indicates that these reduced emissions are equivalent to removing 294,000 vehicles from the road, reducing the oil consumption by nearly 3.5 million barrels, or reducing the gasoline consumption by more than 168 million gallons.

Among other benefits, the biogas produced by anaerobic digesters can be used to generate energy with 41% electrical efficiency. It can also be used as a vehicle fuel with greater efficiency and reduce 95% of carbon dioxide emissions. However, the biogas production process can be not as economically attractive on a large industrial scale as other biofuels. Therefore the optimization of anaerobic digesters process is essential to enhance the productivity of these systems.

In the first section of this chapter a review on the studies that approach the optimization of biogas production though anaerobic digestion using an optimization technique will be presented. Also, other researches that consider the use of a mathematical model to maximize biogas production with different approaches are also presented next.

4.1 Literature review

Some researchers have optimized the production of biogas through anaerobic digestion by modeling its process and applying an optimization technique. For instance, Fakharudin et al., (2003) compared the response surface methodology (RSM) and their proposed model to optimize a biogas production using artificial neural networks and genetic algorithm was made. The structure of the model included two training approaches to feed a set of neural network design. The predictions of the model were used to generate the maximum biogas output and the results were optimized using a genetic algorithm. Their results demonstrated a 0.44% increase of the maximum biogas produced from the RSM

and that the modeling accuracy with low error would not give a better yield. Furthermore, Abu et al., (2010) applied an Artificial Neural Network (ANN) and Genetic Algorithm (GA) to simulate and optimize the digester's biogas production process from a biogas plant. The ANN model was trained, using operational data taken from the plant in a period of 177 days, to simulate the digester operation considering the effect of digester parameters, such as temperature, total solids, total volatile solids, and pH on the biogas yield. The GA was used to optimize and to predict the methane production. The model was validated and proved a 0.87 correlation coefficient of effectiveness to predict the methane production. After finding the optimal operating conditions thru the model the methane production increased by 6.9 %. Moreover, Gueguim et al., (2012) developed an Artificial Neural Network (ANN) to model the biogas production on mixed substrates of saw dust, cow dung, banana stem, rice bran and paper waste. The process of the model was optimized using a Genetic Algorithm (GA) using the model as a fitness function. The data used to train and validate the ANN model were taken from twenty five mini-plot biogas fermentations. A predicted biogas performance of 10.144L was provided using the optimized substrate profile while its evolution gave a biogas production of 10.280L increasing its performance by 8.64%. The optimum biogas production method using these substrates was described in their paper. Most recently, Balmant, et al., (2014) analyzed the optimal residence time and substrate input mass flow rate to maximize the production of methane in anaerobic digestion. The authors used a numerical simulations performed with a general transient mathematical model of an anaerobic biodigester. The three main steps considered for the model were: acidogenesis, acetogenesis and methanogenesis, and was developed for well mixed reactors. Their model describes the transient and steady state system for different operating conditions and biodigester designs. A parametric analysis proved that biogas production is strongly dependent on the polymeric substrate's input and fermentable monomer concentrations, but quite independent of the propionic's input, butyric and acetic acid concentrations. The optimal residence time and substrate input mass flow rate were found by conducting an optimization study and the results showed a sudden dropped of methane from the observed maximum zero, within a 20% range around the optimal operating parameters.

Other researchers have focused their studies to find the optimal batch schedules and residence times in anaerobic digesters to maximize the total gas production. For instance, Curry et al., (1986) considered the problem of scheduling of both single and multiple feedstocks in one digester system were considered. The availability times, biomass quantities, biogas production rates and storage decay were all taken into account to maximize the biogas production during a planned horizon. The feedstock decay while in storage was considered to determine the optimal batch residence times in the digester in order

to achieve the maximum gas production. A dynamic programming algorithm was used to solve the single feedstock batch scheduling problem while a decomposition approach, where the master level allocates time to each feedstock while the sub problems schedule batches within these time allocations, was performed to solve the multiple feedstock problem. Moreover, Deuermeyer et al., (1986) introduced a perishable inventory and production problem related to the production of biogas via anaerobic digestion with a fixed capacity. In their study, a numerical algorithm was provided to optimize the total gas production over a fixed planed horizon. The problem was to determine the optimal residence times for batches in the anaerobic digester considering the biomass decay while in storage. An example problem, based on three different batch sizes with real biomass and digester data, was used to demonstrate the scheduling algorithm and the behavior of the optimal solution. Furthermore, Feldman et al., (1987) proposed a dynamic programming to solve the scheduling of two different types of feedstocks with decreasing production rates. The objective was to maximize the total gas production in a conversion facility of limited capacity. The decision variables considered were the optimal residence times in the digester for both feedstocks and the amount of time the production facility used for digesting the feedstocks. The mathematical procedure as well as an example problem was provided to illustrate the dynamics of the program. More recently, Gim et al., (2001) presented a branch and bound algorithm to solve the scheduling of a single anaerobic digester with multiple feedstocks. The problem was to determine the batch production sequence and the batch residence times in the digester in order to maximize the total gas production considering a specific planning horizon. They also considered the declining viability of stored biomass and the declining rate of methane yield in the digester with time. An example problem was provided to illustrate the solution procedure for three different types of batches with different arrival times.

This chapter will describe the scheduling problem of a single anaerobic digester considering multiple feedstocks with different arrival times. A new Genetic Algorithm was considered to solve the problem and a program was developed in Matlab® to run this optimization technique. The methodology used for the GA as well as the solution obtained will be presented. A sensitivity analysis on the GA's parameters will be provided to achieve maximum gas production and a Design of Experiments was performed to observe which parameters are significant and which are not. Different case studies will be solved to demonstrate both the ability of GA's to solve complex problems and the flexibility of the program developed to solve larger problems.

4.2 Model description

The main problem considered is to determine the sequence in which the feedstocks are going to be processed and the time to allocate each feedstock in the single digester so as to maximize the production of biogas.

In order to calculate the unit gas production of each feedstock given the batch residence time t, the theoretical form of Chynoweth et al., (1981) was employed, which results in the form of

$g_i(t) = \alpha_i\left(1 - e^{(-\beta_i[t-d]^+)}\right)$……..…. (1)

In equation 1, the biomass-gas conversion coefficients of feedstock i are given by α_i and β_i. The setup time, *d*, is included in the batch residence time, therefore the net batch residence time is given by $t - d$. Also, $[t - d]^+$ indicates that only positive values can be considered since there cannot be negative days. Therefore, the maximum of 0 and $t - d$ is taken. It is assumed that the batches of the same feedstock are homogeneous in the digester and that the environment is reasonably constant to keep $g_i(t)$ from varying over time.

Another important factor that affects the biogas production is that stored biomass is subject to decomposition as it is exposed to oxygen. Since only one batch can be process at a time, the remaining batches are stored and wait to be processed. During this period, any stored batch will suffer a declined efficiency that will directly affect its biogas yield. To calculate the decay of each feedstock with respect to the storage time, a decay factor is estimated using the following equation

$h_i(s) = e^{-\gamma_i s}$…….....…… (2)

In equation 2, s is the storage time until the batch is processed in the digester and $\gamma_i$is the storage decay factor of feedstock i.

The actual gas yield of a batch from feedstock *i* that was stored *s* amount of time and whose residence time in the digester is *t* time units is given by the product of the unit gas production and the decay function

$\alpha_i\left(1 - e^{(-\beta_i[t-d]^+)}\right) \cdot e^{-\gamma_i s}$……..…….. (3)

In order to estimate the total gas produced by the anaerobic digester considering different types of feedstocks and arrival times the following mathematical procedure presented by Gim et al., (2001) is used in the present work. In this procedure, the different feedstocks are arranged in increasing order according to their arrival times, assuming that the arrival time for the first feedstock equals to zero, therefore $r_1 = 0$ and $r_i \leq r_j$ if $i < j$. The decision variable $x_{ij} = 1$ specifies the multiple feedstocks

with i being the feedstock type and j the batch number for the denoted feedstock and 0 otherwise and the batch residence time which is represented as t_j for *j* = 1, 2, …., n.

A batch is considered a candidate for the *j*th position if its arrival time (r_i) is less than or equal to its start time, $\sum_{k=1}^{j-1} t_k$. The storage time for the *j*th batch includes the setup time *d* as the decay also occurs during this process, thus $\sum_{k=1}^{j-1} t_k + d - r_i$. Hence, the biogas production of the *j*th batch from feedstock type *i* is expressed as follows.

$$f_{ij}(t_j) = \begin{cases} h_i\left(\sum_{k=1}^{j-1} t_k + d - r_i\right) g_i(t_j) & \text{if } \sum_{k=1}^{j-1} t_k \geq r_i \\ 0 \text{ otherwise} \end{cases} \quad \ldots\ldots (4)$$

The objective function: Maximize $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(t_j) x_{ij}$

Subject to:

$\sum_{j=1}^{n} t_j = T,$ …………………………………………………… (5)

$\sum_{i=1}^{m} x_{ij} = 1, \quad \text{for } j = 1, 2, \ldots, n,$ …………………………… (6)

$\sum_{j=1}^{n} x_{ij} = n_i, \quad \text{for } i = 1, 2, \ldots, m,$ …………………………... (7)

$t_j \geq 0, \quad \text{for } j = 1, 2, \ldots, n,$ ……………………………... (8)

$x_{ij} = 0 \text{ or } 1, \text{for all } i \text{ and } j.$

Where *m* = the number of feedstocks, n_i = the number of batches in feedstock *i*, for *i* = 1, 2, …,*m*, *n* = the total number of batches and T = the planning time horizon.

The first constraint accounts for the total time available and the remaining constraints specify that only one batch can be processed at a time and that all batches are eventually processed.

The objective function is difficult to solve since the decision variables, scheduling sequence and batch residence time, are both at issue. The methodology to determine optimal sequence and residence times to maximize the total gas production given the problem characteristics and constrains is explained next.

4.3 Methodology

A Genetic Algorithm was used to solve the objective function in the problem presented. The steps of the algorithm are as follows.

1. Encoding: The individual's chromosomes include the feedstock sequence and their corresponding residence times as seen in figure 4.1. The first section of the chromosome belongs to the batch sequence which is generated based on the number of batches indicated by the problem. The second half of the chromosome correspond to the residence's time of the

batches in the same order with the restriction that its sum is equal to time allowed to process all the batches.

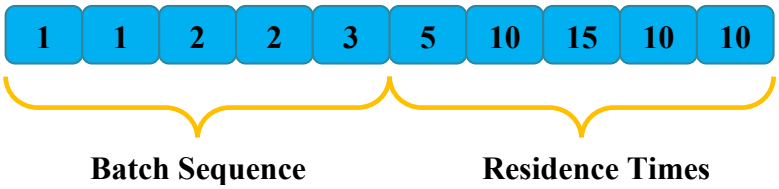


Figure 4.1: Chromosome

2. Initialization: The individuals are randomly generated to form the initial population and each contains a possible solution to the problem.
3. Evaluation: The entire initial population is evaluated according to the fitness function, in this case, the gas production formula mention in section 4.2.

$$f_{ij}(t_j) = \begin{cases} h_i\left(\sum_{k=1}^{j-1} t_k + d - r_i\right) g_i(t_j) & \text{if } \sum_{k=1}^{j-1} t_k \geq r_i \\ 0 \text{ otherwise} \end{cases}$$

After that, the individuals are correspondingly to their fitness value in descending order.

4. Selection: The elite parents are chosen from the best fitted individuals using an elitism rate to pass intact to the next generation. The remaining spots are filled by the tournament selection method considering the entire population. Tournaments are played between two random individuals and the one with the highest gas production is chosen to be parent number 1. Another pair of random individuals is selected and the same criterion is used to select parent number 2.
5. Reproduction: Once the parents are selected they have a specified probability (crossover rate) of being reproduced. Because the order of the chromosome matter a direct swap is not possible. Therefore, if the parents reproduce, the chromosome is split into two and two different single point crossovers are applied to each side to create two children.

   The first point crossover is applied to the batch sequence where the first two columns of "parent 1" and the last three from "parent 2" are taken and create the batch sequence for "child 1". Also, the first two columns of "parent 2" and the last three from "parent 1" are taken and create the batch sequence for "child 2". Similarly, the second point crossover is applied to the residence times where the first two columns of "parent 1" and the last three from "parent 2" create the

residence times for “child 1” and the first two columns of “parent 2” and the last three from “parent 1” create the residence times for “child 2”. Figure 4.2 shows this procedure

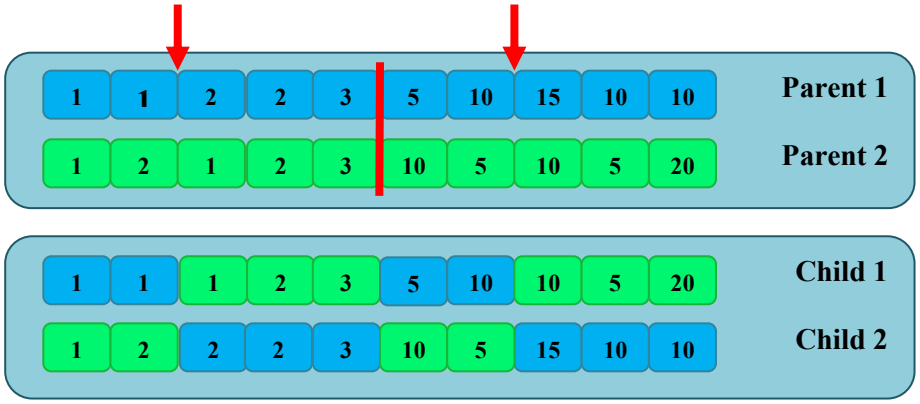


Figure 4.2: First Crossover

Then another set of single point crossovers are applied again to create two more children. This time the first crossover point includes the first three columns from “parent 1” and the last two from “parent 2” to create the batch sequence for “child 3”. Also, the first three columns from “parent 2” and the last two from “parent 1” create the batch sequence for “child 4”. Similarly the second crossover is applied to the residence time where the first three columns from “parent 1” and the last two from “parent 2” create the residence times for “child 3” and the first three columns from “parent 2” and the last two from “parent 1” create the residence times for “child 4”. Figure 4.3 shows this procedure.

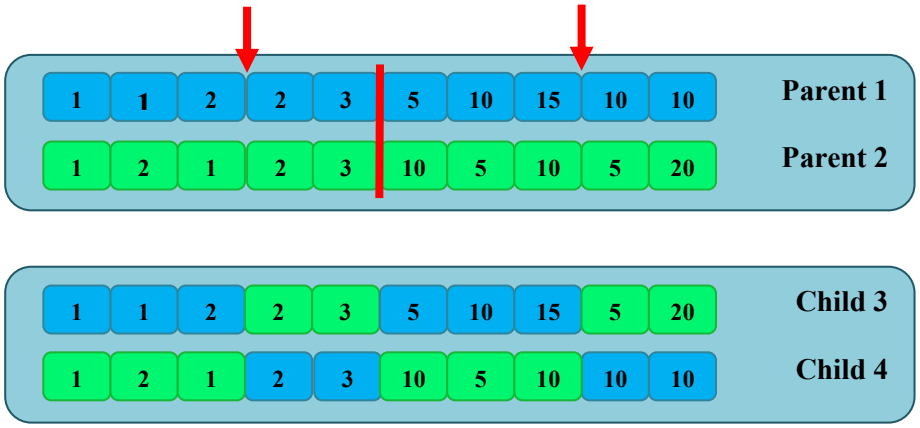


Figure 4.3: Second Crossover

In addition, once the four children are created they have a specified probability of being mutated (mutation rate) which can only be applied to one side of the chromosome or the other. If mutation is

applied, two random points that belong to the same side are swapped. In figure 4.4 the mutation was applied randomly to the batch's sequence. However, the mutation can also be applied to the residence times if the two random points lay on the second half of the chromosome

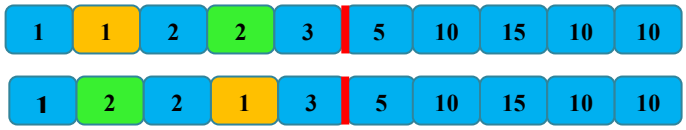


Figure 4.4: Mutation

5. Stopping criterion: The algorithm stops after a specific number of generations and the optimal solution is given by the individual with maximum gas production in the last generation. On the other hand, if the optimal solutions between generations increment by less than an specified epsilon rate then, the algorithm stops and the optimal solution for the problem is given by its last generation.

In Figure 4.5 a flow chart illustrates the process of the algorithm

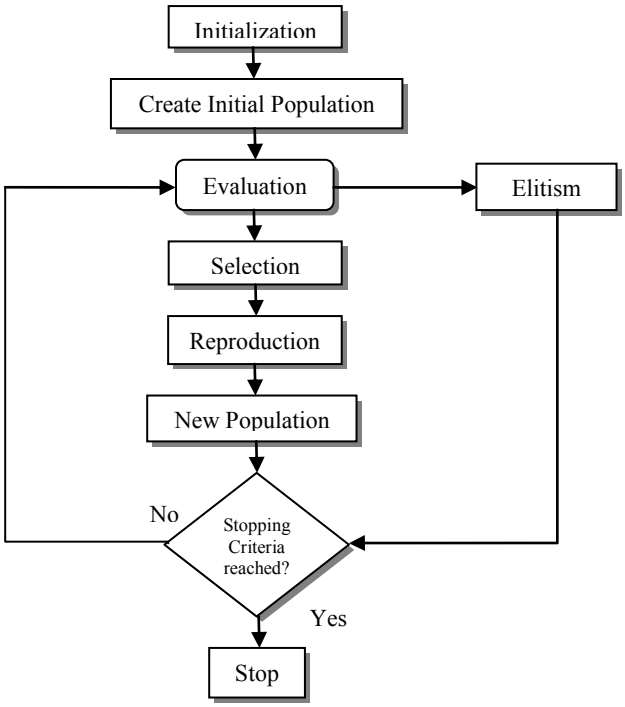


Figure 4.5: Genetic Algorithm Flow Chart

4.4 Program developed

The new Genetic Algorithm was coded in Matlab®. The program consists of four functions, a set of five variables and the main program that runs the GA algorithm. The variables' names are listed in table 4.1.

Table 4.1: Problems' variables

Variables
Alpha: Biomass-gas conversion coefficient for each feedstock
Arrival: Time of arrivals of every feedstock
Batches: Number of batches per feedstock
Beta: Biomass-gas conversion coefficients of feedstocks
Ye: Storage decay factor for each feedstock

The variables contain the parameters and the number of batches for each feedstock type for the example problem described. These variables are crucial to solve the problem, since they contain the information that is required to evaluate the gas production. These variables are called by the functions when the information is required.

The functions' descriptions are presented as follow

- GA

The GA function starts the algorithm previously described. The problems' parameters such as the total time (T), setup time (d) and the time increment (MTB) for the residence times are also defined in this function. The GA's parameters including population size, number of generations, epsilon to end the algorithm, elitism rate, crossover rate and mutation rate are also found here. Both the problems' parameters and the GA's parameters are shown in table 4.2.

Table 4.2: GA and Problems' Parameters

GA's Parameters	
Population size	50
Number of generations	20
Epsilon rate	0.1
Elitism rate	0.25
Crossover rate	0.75
Mutation rate	0.01
Problem Parameters	
Total time period	50
Setup time	1
Time intervals	5

- P_gent

This function creates the initial population. It generates 50 random individuals that consist of ten columns each. The first five columns refer to the sequence and the last five to the residence times. The sequences are generated based on the information provided by the variables previously mention. The residence times are ranged from 0 to 50 with increments of fives with the only restriction that the sum has to add up to 50 as the time horizon for this problem was set to 50 days which will be later explained.

- Eval_seq2

This function contains the gas production formula to evaluate the individuals from the initial population and individuals from subsequent generations. The gas production function is as follows:

$$f_{ij}(t_j) = h_i(\sum_{k=1}^{j-1} t_k + d - r_i)g_i(t_j) \qquad if \sum_{k=1}^{j-1} t_k \geq r_i$$

$$maximize \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}(t_j)x_{ij}$$

- Reproduction

This function sort the results evaluated in eval_seq2 in descended order and chooses the elite parents. It also defines how many children are needed depending on how many elite parents are chosen for the next generation. The parent selection is by tournaments and the reproduction uses crossover and mutation defined in the GA function.

- Check

This function corrects the chromosomes that were generated by reproduction. Those children which did not satisfy the problems criteria after they were created through crossover and mutation are modified to fit the restrictions. For instance, if a "child" sequence has more batches than those available in the problem this function chooses one of the extra batches randomly and replaces it with a batch that was not in the sequence. In the same way, if the residence times exceed the allowable time, this function chooses one randomly and subtracts the exceeding time excluding those with zero time.

In addition to the variables and functions described, there are extra variables that save information to have easy access if need it. Some of these variables contain information such as optimal solution and the algorithms' progress. The Matlab® Code can be found in Appendix A.

4.5 Case Study 1

An example problem taken from Gim et al., 2001 was used to solve the scheduling of a single anaerobic digester with multiple feedstocks. The problem considers three different types of feedstocks i with one or two batches n_i and different arrival times r_i. All batches need to be processed in the digester in a time period of 50 days with the restriction that only one batch can be processed at a time. The batch set up time d is one day. The feedstock parameters are shown in the table 4.3 below.

Table 4.3: Feedstock Parameters Case Study 1

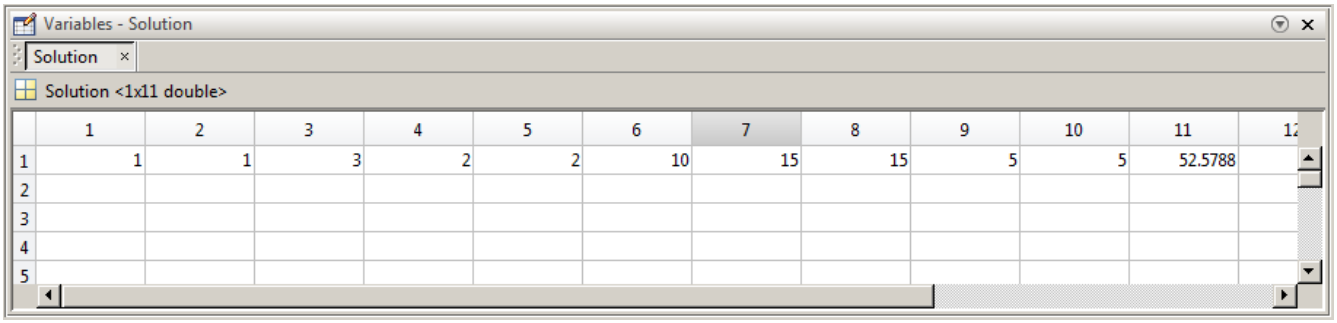
Feedstock	$g_i(t)$		$h_i(s)$	Arrival times	Batches
i	α_i	β_i	γ_i	r_i	n_i
1	28	0.1	0.021	0	2
2	13.2	0.09	0.015	15	2
3	18	0.12	0.02	25	1

As a side note, it was observed that the data from the original example problem taken from Gim el al., 2001 contains a typographical error as the parameter β_i from feedstock 1 was stated to be 1.0 instead of 0.1. Their results were not affected by this error since the correct parameter was used. After making this observation, the proper parameter value $\beta_i = 0.1$ was also employed in this procedure.

The scheduling of a single anaerobic digester with multiple feedstocks' problem described was solved using a Genetic Algorithm programmed in Matlab®. The optimal solution given in the first run was < 1 1 3 2 2 10 15 15 5 5 > with a total gas production of 52.5787. After running the program

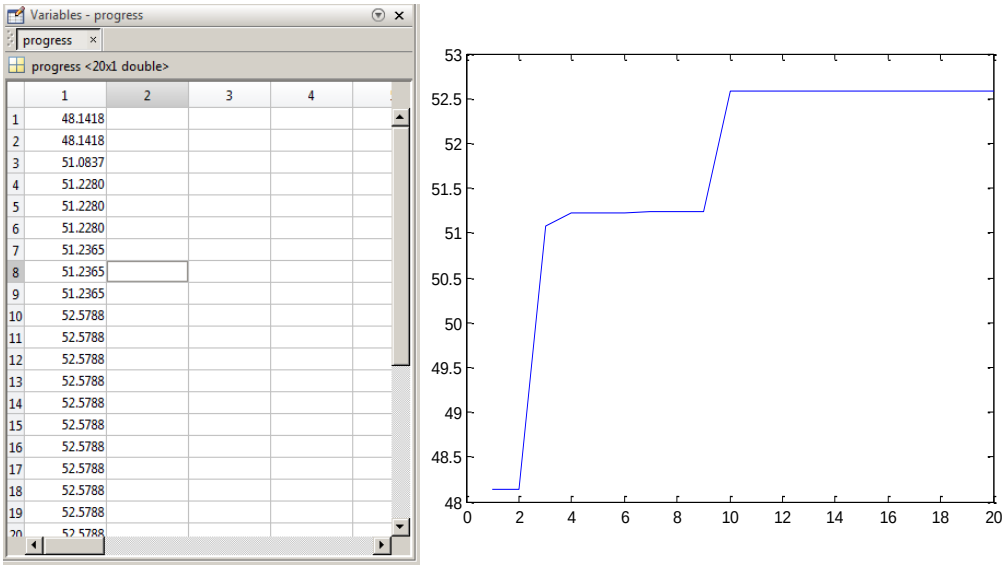
different times the same optimal sequence < 1 1 3 2 2 > is obtained but the residence times are different with each run, making the solution vary slightly. The total computational time was 3.563 seconds using an Acer computer with a processor Intel® Celeron® CPU 900 @ 2.20 GHz 2.19 GHz.

Figure 4.6, is the Matlab® output for the optimal sequence and optimal residence times corresponding to the maximum production of gas found. Figure 4.7, shows the output for the optimal solutions through each generation finding the global optima in generation 10 and the graph illustrates how the solutions are evolving through each generation. Finally figure 4.8 shows the optimal solution for the case study.



	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	3	2	2	10	15	15	5	5	52.5788	
2												
3												
4												
5												

Figure 4.6: Optimal solution output for case study 1



	1	2	3	4
1	48.1418			
2	48.1418			
3	51.0837			
4	51.2280			
5	51.2280			
6	51.2280			
7	51.2365			
8	51.2365			
9	51.2365			
10	52.5788			
11	52.5788			
12	52.5788			
13	52.5788			
14	52.5788			
15	52.5788			
16	52.5788			
17	52.5788			
18	52.5788			
19	52.5788			
20	52.5788			

Figure 4.7: Algorithms' Evolution

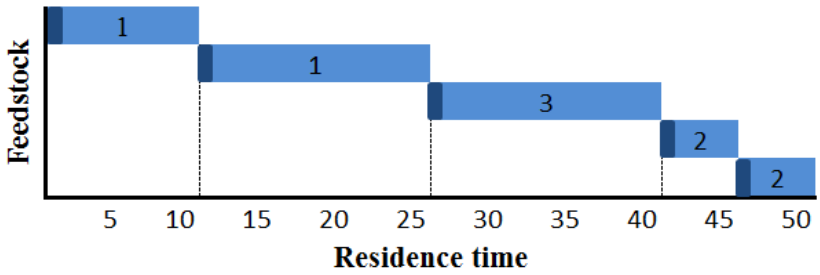


Figure 4.8: Optimal Solution Case Study 1

The following is a representation of the example problem described and the optimal solution obtained using a GA. Illustration 4.1 shows two batches of the same kind arriving at time zero to be processed in one anaerobic digester. Since only one batch can be processed at a time, one of the batches is goes in the digester and is processed for ten days while the other is put on hold in storage. During this time, the stored batch composition will decay as it is being exposed to oxygen. In illustration 4.2, after 10 days the stored batch goes into the digester for processing. While it is being processed, two more batches of different type of feedstock arrive to the facility at time 15. Because the digester is busy, the two batches go to storage and wait there to be schedule. In illustration 4.3, one single batch belonging to a third kind of feedstock arrives to the facility at time 25. Since the digester is available, the batch goes into the digester and is processed for 15 days. Meanwhile, the previous two batches of feedstock 2 remain in storage. In illustration 4.4 one of the stored batches is processed for 5 days at time 40 and the other one remains stored. Finally, in illustration 4.5 the last batch belonging to feedstock 2 is processed in the digester for the remaining time available and the total gas yield is 52.5787.

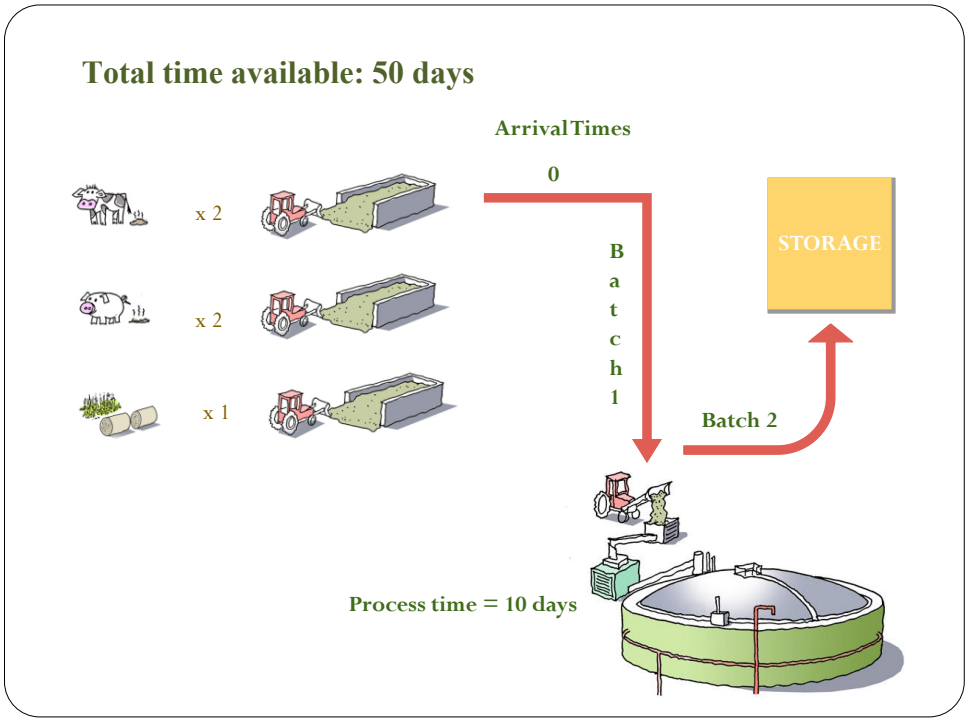


Illustration 4.1: Anaerobic digester at time 0.

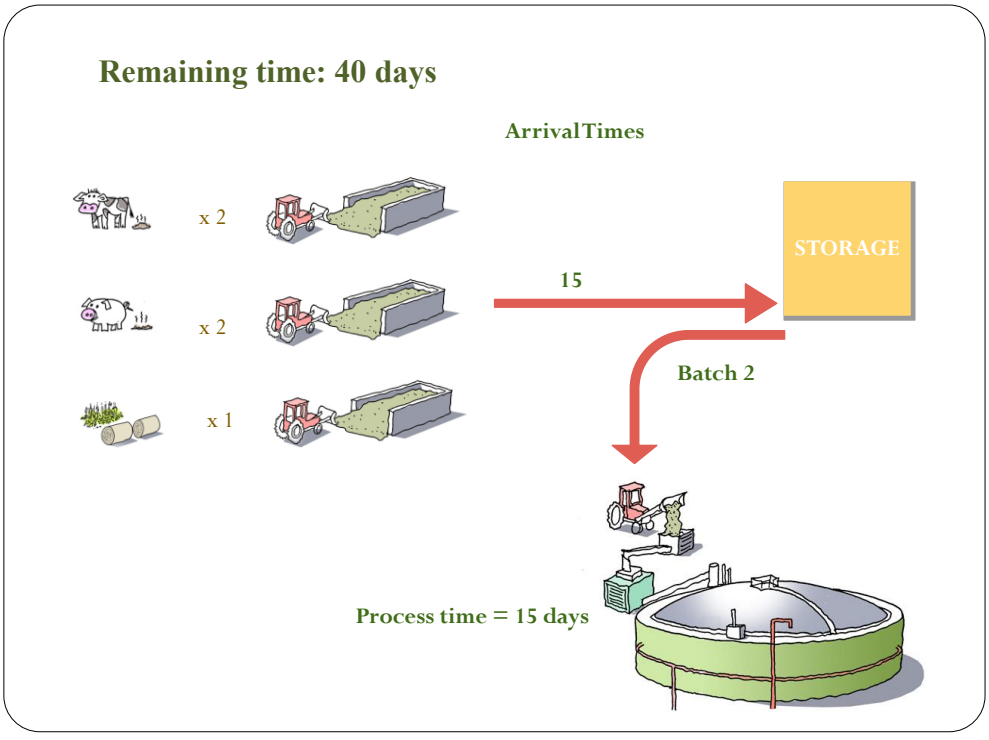


Illustration 4.2: Anaerobic digester at time 10

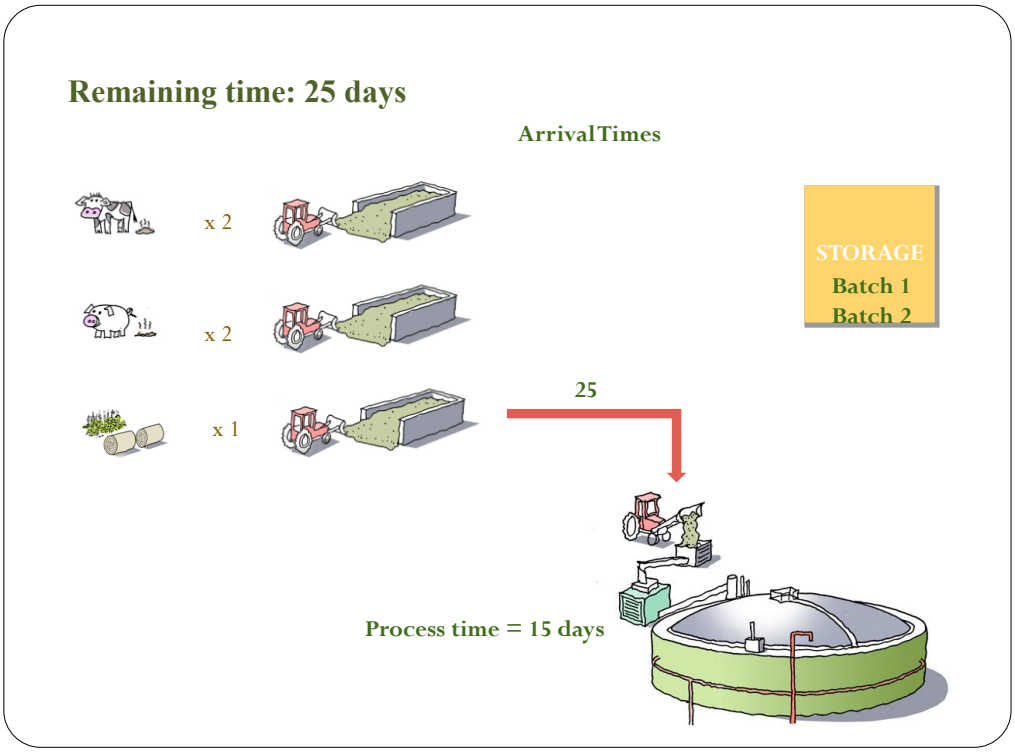


Illustration 4.3: Anaerobic digester at time 25

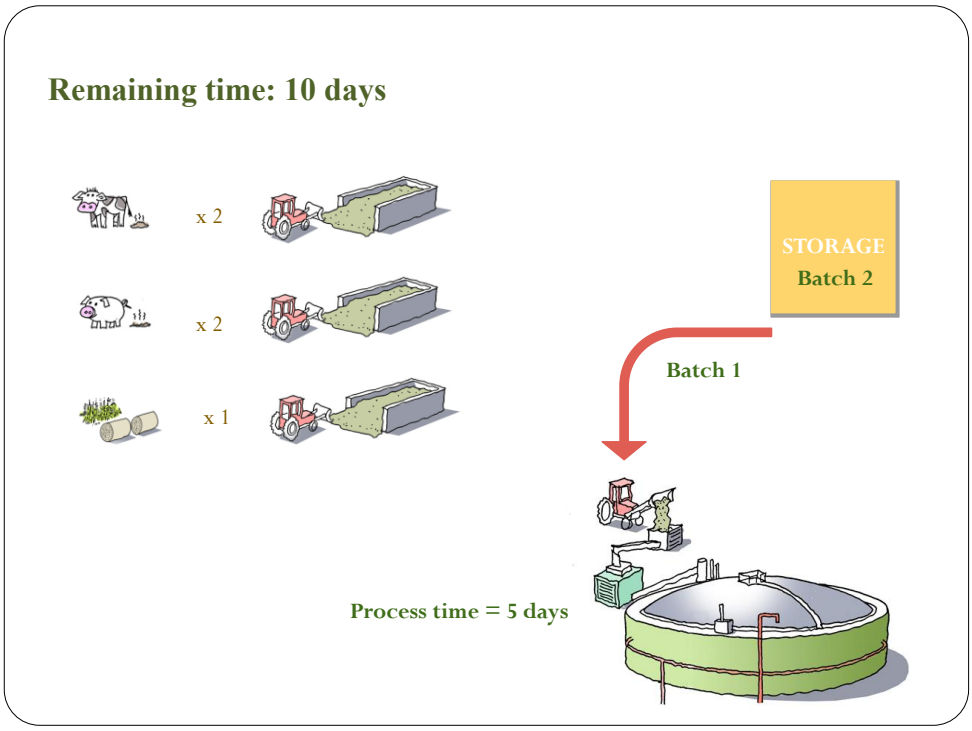


Illustration 4.4: Anaerobic digester at time 40

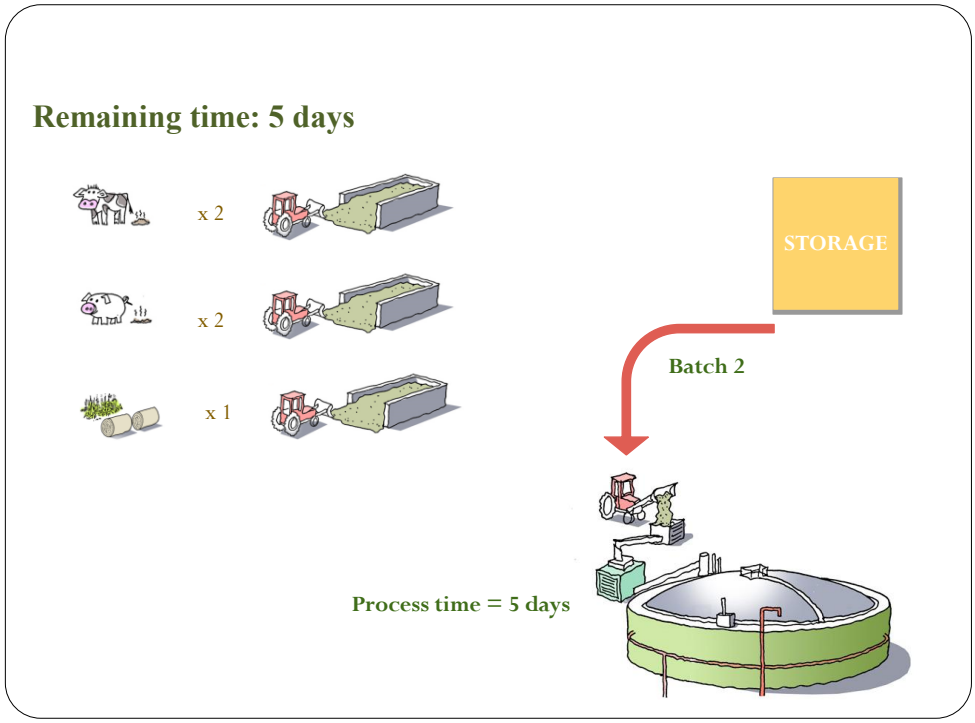


Illustration 4.5: Anaerobic digester at time 45

4.5 Comparison with previous results

When solving this problem using a branch and bound algorithm with dynamic programming as described in Gim et al., (2001), the partial sequence that leads to the sequence < 1 1 3 2 2> (obtained as the optimal solution through the GA) was fathom, which turns to have better gas production with residence times < 10 15 15 5 5 > respectively. The optimal solution found using their proposed method as shown in the table 4.4.

Table 4.4: Branch and bound algorithm's optimal solution

Iteration	Node	Upper bounds	Residence times
1	⟨1,1,2,2,3⟩	51.07	$t_1 = 10, T_2 = 10, T_3 = 5, T_4 = 10, T_5 = 15$
2	⟨1⟩	55.54	$t = 10, T_1 = 15, T_2 = 15, T_3 = 10$
3	⟨2⟩	37.32	$t = 15, T_1 = 20, T_2 = 5, T_3 = 10$
4	⟨3⟩	23.25	$t = 35, T_1 = 15, T_2 = 0, T_3 = 0$
5	⟨1,1⟩	53.97	$t = 20, T_1 = 0, T_2 = 15, T_3 = 15$
6	⟨1,2⟩	53.87	$t = 20, T_1 = 10, T_2 = 10, T_3 = 10$
7	⟨1,3⟩	47.29	$t = 35, T_1 = 5, T_2 = 10, T_3 = 0$
8	⟨1,1,2⟩	53.97	$t = 25, T_1 = 0, T_2 = 10, T_3 = 15$
9	⟨1,1,3⟩	51.85	$t = 40, T_1 = 0, T_2 = 10, T_3 = 0$
10	⟨1,1,2,3,2⟩	52.42	$t_1 = 10, t_2 = 10, t_3 = 5, t_4 = 15, t_5 = 10$
11	⟨1,2,1⟩	51.80	$t = 30, T_1 = 0, T_2 = 10, T_3 = 10$
12	⟨1,2,2⟩	52.52	$t = 25, T_1 = 10, T_2 = 0, T_3 = 15$
13	⟨1,2,3⟩	50.24	$t = 35, T_1 = 10, T_2 = 5, T_3 = 0$
14	⟨1,2,2,1,3⟩	49.62	$t_1 = 15, t_2 = 5, t_3 = 5, t_4 = 10, t_5 = 15$
15	⟨1,2,2,3,1⟩	49.79	$t_1 = 15, t_2 = 5, t_3 = 5, t_4 = 10, t_5 = 15$

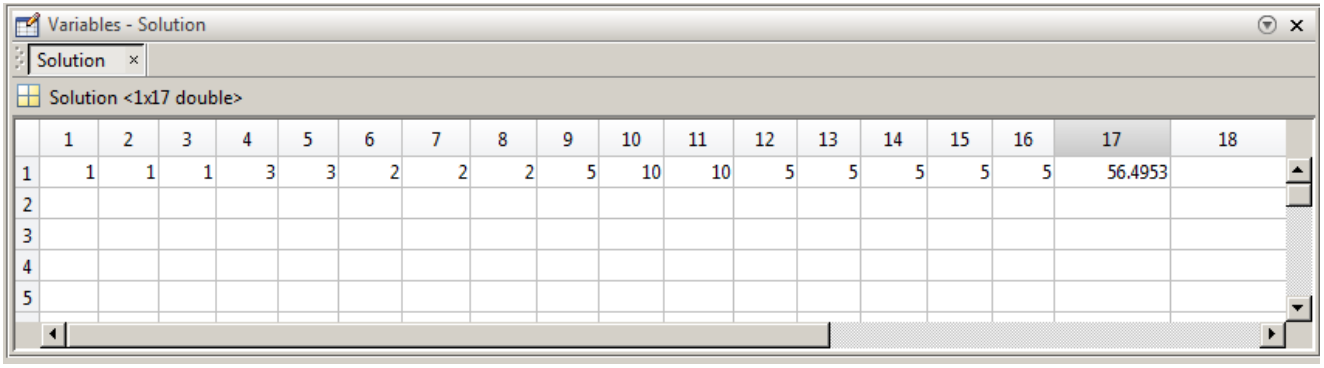
4.6 Case Study 2

To evaluate the capacity of the developed algorithm to solve larger problems, the GA is applied to a case study in which the number of batches for each feedstock is increased. The rest of the parameters, both the GA's and the problems', where kept the same. Table 4.5 show the new set of batches for each feedstock type with their corresponding parameters.

Table 4.5: Feedstock Parameters Case Study 2

Feedstock	$g_i(t)$		$h_i(s)$	Arrival times	Batches
i	α_i	β_i	γ_i	r_i	n_i
1	28	0.1	0.021	0	3
2	13.2	0.09	0.015	15	3
3	18	0.12	0.02	25	2

The optimal sequence for this case study using the new Genetic Algorithm is <1 1 1 3 3 2 2 2> with residence times <5 10 10 5 5 5 5 5> respectively. The computational time was 1.905 seconds. Figure 4.9, is the Matlab® output for the optimal sequence and optimal residence times corresponding to the maximum production of gas found. Figure 4.10, illustrates how the solutions are evolving through each generation. Finally figure 4.11 shows the optimal solution for the case study.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	1	3	3	2	2	2	5	10	10	5	5	5	5	5	56.4953	
2																		
3																		
4																		
5																		

Figure 4.9: Optimal solution output for case study 2

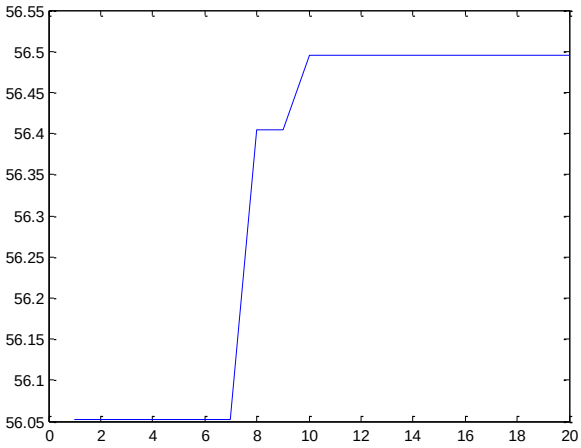


Figure 4.10: Evolution of the algorithm case study 2

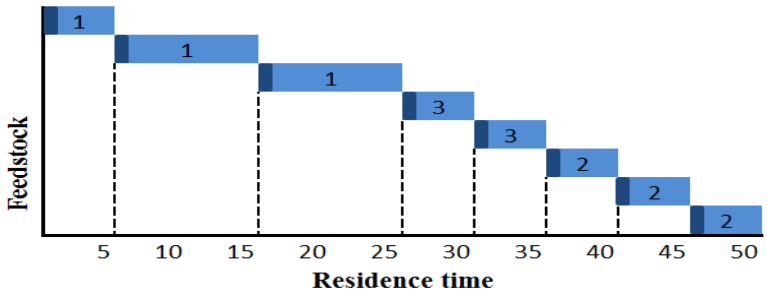


Figure 4.11: Optimal solution case study 2

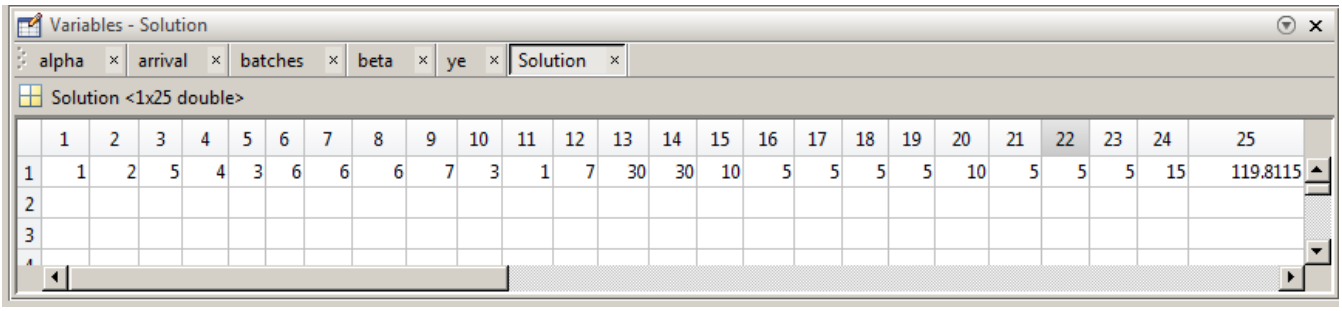
4.6 Case Study 3

A larger problem containing more feedstocks is considered in case study 3. The time to process all the batches (T) was changed to 130 days, the population size (p_size) was increased to 100 individuals and the number of generations to 150 while the rest of the GA's parameters were kept the same. The problem's parameters α, β, γ, r and n are shown in table 4.6.

Table 4.6: Feedstock Parameters Case Study 3

Feedstock	$g_i(t)$		$h_i(s)$	Arrival times	Batches
i	α_i	β_i	γ_i	r_i	n_i
1	20	0.1	0.015	0	2
2	28	0.08	0.018	30	1
3	15	0.13	0.021	45	2
4	13	0.21	0.019	50	1
5	25	0.09	0.025	60	1
6	18	0.2	0.02	80	3
7	21	0.11	0.011	100	2

The optimal sequence found is <1 2 5 4 3 6 6 6 7 3 1 7> with residence times <30 30 10 5 5 5 5 10 5 5 5 15> achieving a total gas production of 119.8115. The computational time to solve this problem was 32.217 seconds. Figure 4.12, is the Matlab® output for the optimal sequence and optimal residence times corresponding to the maximum production of gas found. Figure 4.13, illustrates how the solutions are evolving through each generation. Finally figure 4.14 shows the optimal solution for the case study.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	2	5	4	3	6	6	6	7	3	1	7	30	30	10	5	5	5	5	10	5	5	5	15	119.8115
2																									
3																									

Figure 4.12: Optimal solution output for example 3.

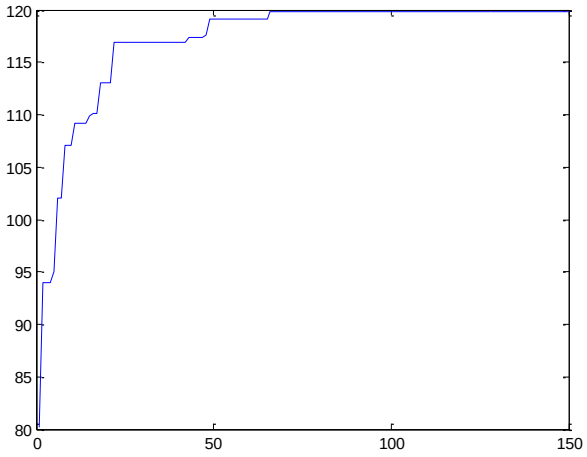


Figure 4.13: Evolution of the algorithm in example 3.

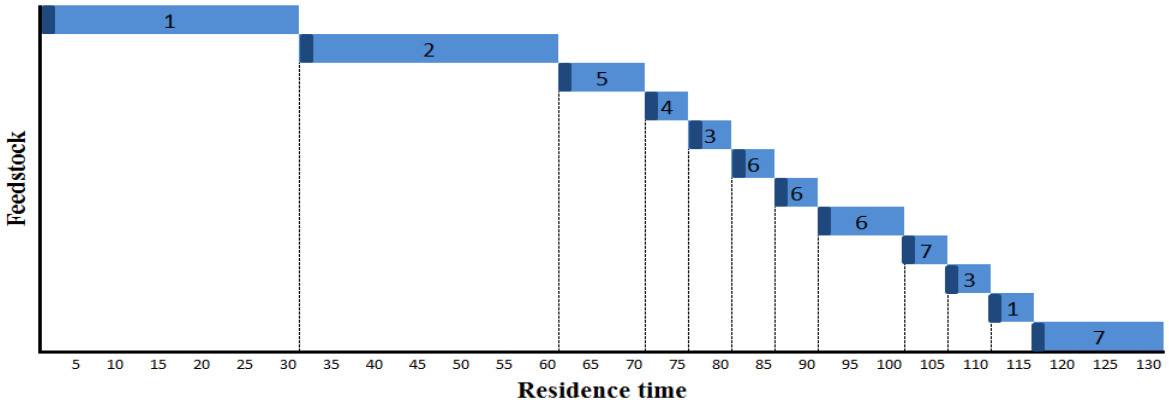


Figure 4.14: Optimal solution for case study 3

Conclusion

Very little research regarding the scheduling of anaerobic digesters can be found. Therefore, there is a vast opportunity to expand these studies. In this chapter, it was proved that Genetic algorithms can be used to optimize the production of biogas by finding the best sequence in which batches need to be processed in a digester as well as their processing time. The problem stated in Gim et al., (2001) which was solved by the branch and bound algorithm using dynamic programming, was used and the results where compared. Even though the optimal solutions found by each of the techniques do not vary that much, the optimal sequence found by the GA was fathom by the branch and bound as not an optimal solution. The difference between results may not be significant in such a small problem, but when more feedstocks and more batches are considered the solutions may vary a lot more.

Chapter 5: Sensitivity Analysis

5.1 Introduction

Genetic algorithms have been applied in many different fields to solve complex optimization problems. Researchers have mentioned the impact that the different GA's parameters such as the population size, mutation rate, elitism rate and number of generations have in the final solution found. In the present work, a sensitivity analysis was performed to evaluate different parameters for the first case study presented in the previous chapter and to test their performance on the total gas production obtained by the optimal solution and the time the program developed takes in finding the optimal solution. In addition to the GA's parameters, the allowable time increment for the batches' residence times in the digester was also considered for the sensitivity analysis to examine the impact on the total gas production when this parameter is changed as well as the computational time. Moreover, a fractional factorial design was done to evaluate the significance of these parameters and their effects in finding the optimal solution and to provide a recommended value for all of them.

The parameters considered for the analysis are: Population size, number of generations, elitism, crossover and mutation.

The parameters considered initially where:

Population size = 50

Number of generations = 20

Elitism = 25%

Crossover = 75%

Mutation = 1%

The number of Population size was increased by fives to have a bigger range of solutions and increase the probability of finding best fitted individuals. The number of generations was also increased by fives in every analysis to give the algorithm a better opportunity to evolve and explore different solutions. Elitism was increased by 1% to keep the best fitted individuals in subsequent generations and to evaluate its impact. Finally, mutation was again increased by 1% to see if by altering the chromosomes the results improve or worsen.

The sensitivity analysis was performed to evaluate both the computational time of the program and the maximum gas production. Table 5.1 shows the results of the analysis. It was observed that the maximum gas production obtained was the same from generation 60 through 90, although the computational time varied every time. Using the minimum computational time as a decision criteria, the optimal solution is given when the parameters are equal to population size 60, generations 30, elitism

27%, crossover 77% and mutation 1.2% with a maximum gas production of 52.57877 and a computational time of 3.152 seconds.

Table 5.1: Sensitivity analysis-five days increments.

MTB	Population Size	generations	Elite	Crossover %	Mutation %	Computational Time	Maximum Gas Production	Generation
5	50	20	0.25	75	1	2.666 s	52.54054322	20
5	55	25	0.26	76	1.1	2.800 s	52.54054322	12
5	60	30	0.27	77	1.2	3.152 s	52.57877749	24
5	65	35	0.28	78	1.3	3.428 s	52.57877749	16
5	70	40	0.29	79	1.4	3.674 s	52.57877749	33
5	75	45	0.3	80	1.5	4.057 s	52.57877749	11
5	80	50	0.31	81	1.6	4.325 s	52.57877749	17
5	85	55	0.32	82	1.7	3.591 s	52.57877749	21
5	90	60	0.33	83	1.8	5.129 s	52.57877749	10
5	95	65	0.34	84	1.9	5.570 s	52.54054322	21
5	100	70	0.35	85	2	6.165 s	52.54054322	23

Because the original problem was solved using only increments of five days for the residence times another sensitivity analysis was done exploring increments of one day. The reason for this study was to explore all possible solutions and obtain a more realistic case study. The same parameters described were analyzed and table 5.2 shows the results. The maximum gas production is obtained when the parameters are equal to population size 90, generations 60, elitism 33%, crossover 83% and mutation 1.8% yielding 53.0663 of gas with a computational time of 5.767 seconds.

Table 5.2: Sensitivity analysis-one day increment.

MTB	Population Size	generations	Elite	Crossover %	Mutation %	Computational Time	Maximum Gas Production	Generation
1	50	20	0.25	75	1	2.605 s	52.27409246	7
1	55	25	0.26	76	1.1	2.848 s	52.65022698	12
1	60	30	0.27	77	1.2	3.726 s	52.54996795	26
1	65	35	0.28	78	1.3	3.621 s	52.74557102	30
1	70	40	0.29	79	1.4	4.059 s	52.82949091	19
1	75	45	0.3	80	1.5	4.436 s	53.06440505	42
1	80	50	0.31	81	1.6	4.531 s	52.55578988	41
1	85	55	0.32	82	1.7	5.437 s	52.54516956	37
1	90	60	0.33	83	1.8	5.767 s	53.06634228	41
1	95	65	0.34	84	1.9	6.187 s	52.54516956	26
1	100	70	0.35	85	2	7.068 s	52.86893156	70

5.2 Design of Experiments

A design of experiments was done in Minitab® to determine if the parameters chosen for the GA affect both the computational time and the total gas produced given by the optimal solution found. The parameters studied were: Time increments (MTB), population size (PZISE), number of generations, epsilon rate, crossover rate and mutation rate. Therefore a two level factorial design was created with 32

runs and resolution IV. The values for each factor considered are shown in the table 5.3 and the factor's combination and responses are shown in table 5.4.

Table 5.3: Factors levels.

Parameter	Low	Hig
MTB	1	5
PSIZE	40	60
Generation	10	30
Epsilon	0.09	0.11
Elitism	0.1	0.4
Crossover	0.6	0.9
Mutation	0.001	0.02

Table 5.4: Factor's combinations and responses

MTB	PZISE	Generations	Epsilon	Elitism	Crossover	Mutation	Time	Production
1	40	10	0.09	0.1	0.9	0.02	1.301	51.2584883
5	40	10	0.09	0.1	0.6	0.001	0.692	51.5675211
1	60	10	0.09	0.1	0.6	0.001	3.697	51.6201307
5	60	10	0.09	0.1	0.9	0.02	1.427	52.5787775
1	40	30	0.09	0.1	0.6	0.02	2.021	52.5151999
5	40	30	0.09	0.1	0.9	0.001	2.77	52.5405432
1	60	30	0.09	0.1	0.9	0.001	1.693	52.8654074
5	60	30	0.09	0.1	0.6	0.02	1.061	52.5787775
1	40	10	0.11	0.1	0.6	0.001	1.285	51.6182851
5	40	10	0.11	0.1	0.9	0.02	1.465	52.423425
1	60	10	0.11	0.1	0.9	0.02	1.539	52.3492233
5	60	10	0.11	0.1	0.6	0.001	1.087	52.3315709
1	40	30	0.11	0.1	0.9	0.001	2.587	52.4236013
5	40	30	0.11	0.1	0.6	0.02	2.334	52.5212897
1	60	30	0.11	0.1	0.6	0.02	1.47	52.4843478
5	60	30	0.11	0.1	0.9	0.001	3.576	52.5405432
1	40	10	0.09	0.4	0.9	0.001	0.541	51.0113117
5	40	10	0.09	0.4	0.6	0.02	2.217	52.0374947
1	60	10	0.09	0.4	0.6	0.02	1.272	51.7635936
5	60	10	0.09	0.4	0.9	0.001	1.642	52.5212897
1	40	30	0.09	0.4	0.6	0.001	1.273	52.6959049
5	40	30	0.09	0.4	0.9	0.02	2.459	52.5787775
1	60	30	0.09	0.4	0.9	0.02	1.463	52.6847328
5	60	30	0.09	0.4	0.6	0.001	5.014	52.3315709
1	40	10	0.11	0.4	0.6	0.02	0.62	51.3746
5	40	10	0.11	0.4	0.9	0.001	1.714	51.8477759
1	60	10	0.11	0.4	0.9	0.001	1.097	52.324806
5	60	10	0.11	0.4	0.6	0.02	1.671	52.423425
1	40	30	0.11	0.4	0.9	0.02	1.692	52.5348683
5	40	30	0.11	0.4	0.6	0.001	1.72	52.5405432
1	60	30	0.11	0.4	0.6	0.001	2.133	52.4786442
5	60	30	0.11	0.4	0.9	0.02	2.22	52.5787775

5.3 Results

After analyzing the ¼ factorial designs the only factor that is significant to the computational time is the number of generations allowed to run the algorithm. The factors that are not significant to the computational time include: Time increment, population size, epsilon rate, elitism rate, crossover rate and mutation rate. The normal plot of the effects corresponding to the computational time is shown in figure 5.1.

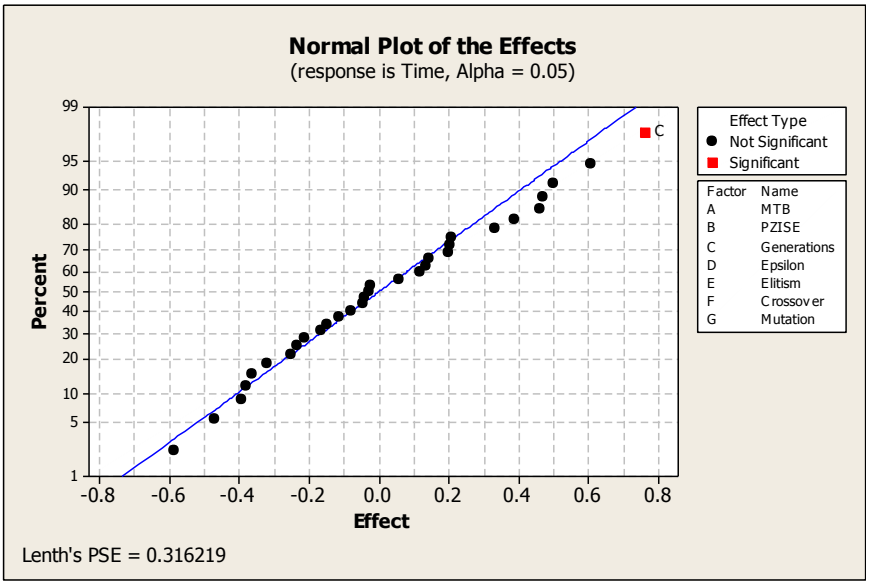


Figure 5.1: Normal plot of the effects for computational time

On the other hand, the factors that are significant to the total gas production include: the time increment for the residence times, initial population size and number of generations. Also, the iterations time increment*generations, population size*generations, generations*epsilon and population size*crossover. The factors that are not significant to the gas production are elitism rate and mutation rate. The normal plot of the effects corresponding to the total gas production is shown in figure 5.2.

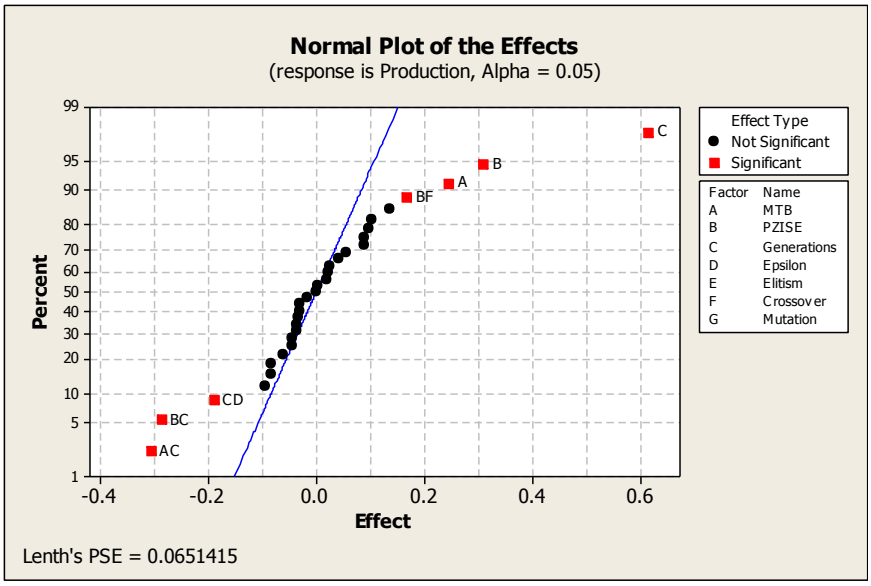


Figure 5.2: Normal plot of the effects for gas production

After concluding which factors are significant and which factors are not, the model was reduced taking away the non significant factors and their corresponding iterations. The factor that is significant to the computational time remains the same while another significant factor for the gas production was found which corresponds to mutation rate. Figures 5.3 and 5.4 show the normal plot of the standardized effects for computational time and total gas productions respectively.

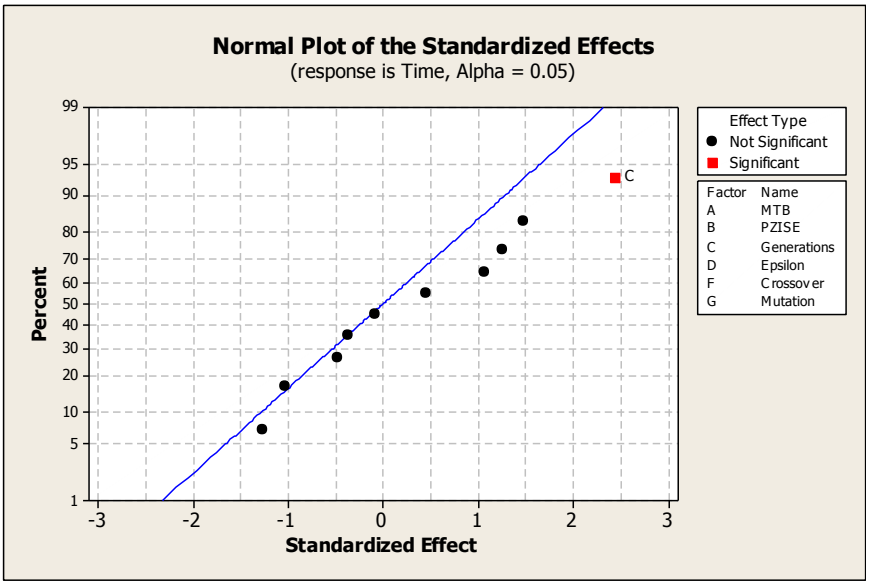


Figure 5.3: Normal plot of the standardized effects for computational time

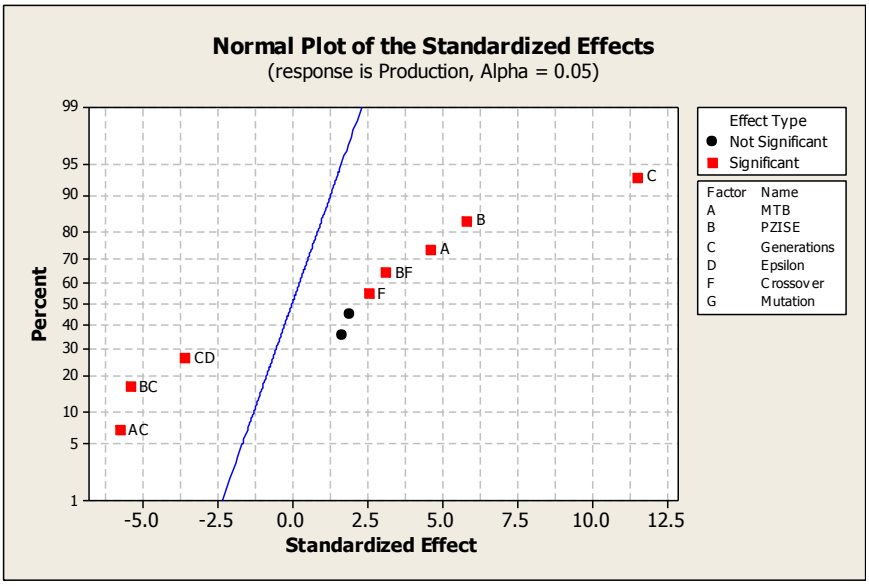


Figure 5.4: Normal plot of the standardized effects for gas production.

The main effects plot is a graphical representation of the factors that are significant to the response variables which were found during the design's analysis. The effect that is significant to the computational time is represented in figure 5.5 which basically describes how the means increment as the number of generations increases.

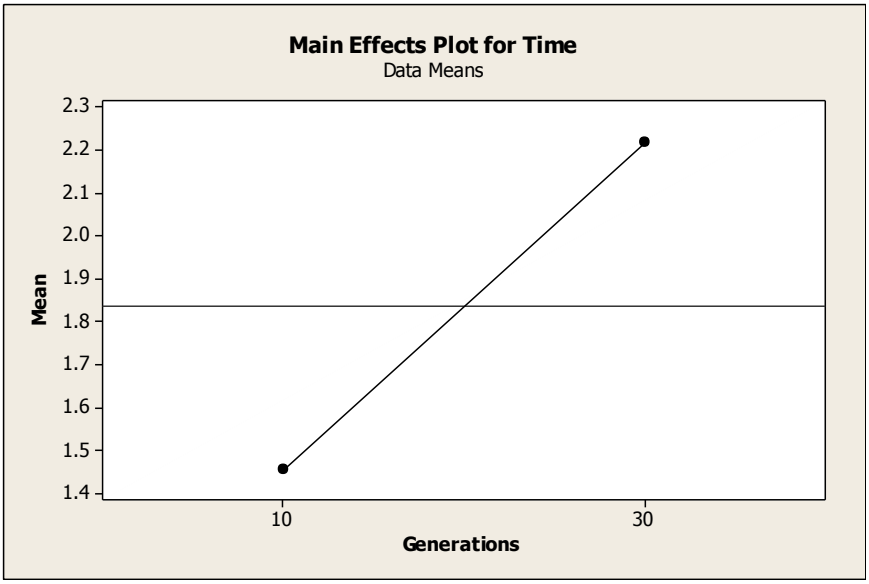


Figure 5.5: Main effects plot for computational time.

Finally, the data means for the gas production response were graphed in a cube plot to determine which factors are optimal for the developed GA to find better solutions therefore higher production of

gas (Figure 5.6). The highest mean 52.7751 is obtained when the crossover rate is equal to 0.9, the number of generations is 30, the initial population size is 60, the epsilon rate is 0.09 and the time is incremented by one day.

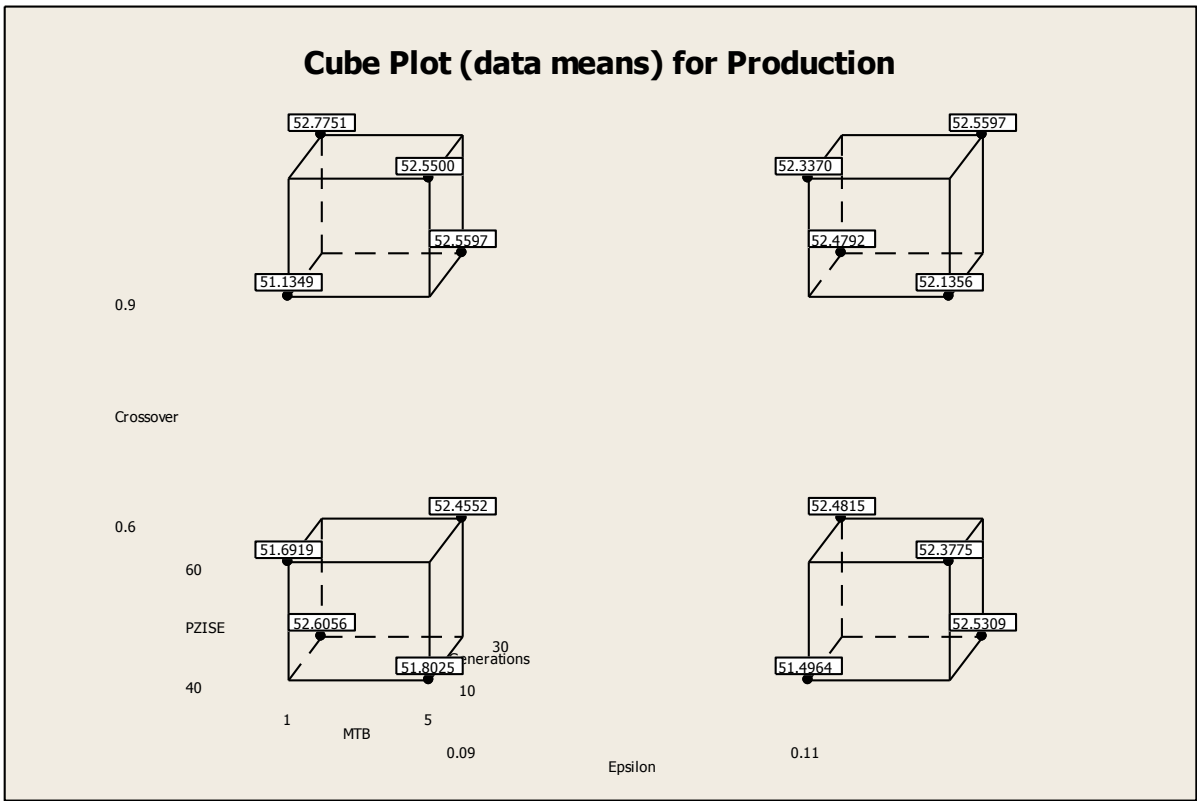


Figure 5.6: Cube plot for gas production.

In the analysis of variance for the computational time, we obtain a P-value of 0.023 with and F-value 6.02 corresponding to the "number of generations" factor indicating that the means are significantly different. Assuming a significance level of 0.05 this factor is the only one that shows to have a difference in means as seen in figure 5.7. On the other hand, the analysis of variance for the production of gas we obtain a P-value of 0.000 and F-value 21.43 that correspond to the "time increment" factor, a P-value of 0.000 with F-value of 34.06 that correspond to the "population size" factor, a P-value of 0.000 with F-value of 133.80 corresponding to the "number of generations" factor and a P-value of 0.018 with F-value of 6.56. Also, assuming a significance level of 0.05 these factors are the ones that prove to have main effects as seen in figure 5.8.

Analysis of Variance for Time (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	6	8.6822	8.6822	1.44703	1.87	0.135
MTB	1	1.7043	1.7043	1.70432	2.20	0.153
PZISE	1	0.9015	0.9015	0.90149	1.16	0.293
Generations	1	4.6657	4.6657	4.66575	6.02	0.023
Epsilon	1	0.1701	0.1701	0.17009	0.22	0.644
Crossover	1	0.0045	0.0045	0.00454	0.01	0.940
Mutation	1	1.2360	1.2360	1.23599	1.59	0.221
2-Way Interactions	4	2.3094	2.3094	0.57735	0.74	0.572
MTB*Generations	1	1.2242	1.2242	1.22422	1.58	0.223
PZISE*Generations	1	0.1039	0.1039	0.10385	0.13	0.718
PZISE*Crossover	1	0.8176	0.8176	0.81760	1.05	0.316
Generations*Epsilon	1	0.1637	0.1637	0.16374	0.21	0.651
Residual Error	21	16.2842	16.2842	0.77544		
Total	31	27.2758				

Figure 5.7: Analysis of variance for computational time.

Analysis of Variance for Production (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	6	4.57913	4.57913	0.76319	33.73	0.000
MTB	1	0.48486	0.48486	0.48486	21.43	0.000
PZISE	1	0.77066	0.77066	0.77066	34.06	0.000
Generations	1	3.02691	3.02691	3.02691	133.80	0.000
Epsilon	1	0.08469	0.08469	0.08469	3.74	0.067
Crossover	1	0.14844	0.14844	0.14844	6.56	0.018
Mutation	1	0.06358	0.06358	0.06358	2.81	0.108
2-Way Interactions	4	1.91166	1.91166	0.47791	21.12	0.000
MTB*Generations	1	0.74503	0.74503	0.74503	32.93	0.000
PZISE*Generations	1	0.65604	0.65604	0.65604	29.00	0.000
PZISE*Crossover	1	0.22504	0.22504	0.22504	9.95	0.005
Generations*Epsilon	1	0.28554	0.28554	0.28554	12.62	0.002
Residual Error	21	0.47509	0.47509	0.02262		
Total	31	6.96587				

Figure 5.8: Analysis of variance for gas production.

Conclusion

In this chapter, a sensitivity analysis was made to evaluate how the different parameters of the GA affect the quality of the final solution. Moreover, to find the different parameters of the problem in the GA the data means for the gas production response were graphed in a cube plot to obtain the best values for the different GA parameters to be able to obtain an optimal solution in a less computational time.

It was observed that the best parameters are: population size = 50, number of generations = 30, elite rate = 27%, crossover rate = 77% and mutation rate = 1.2% for the case where five days increments are considered for scheduling the batches. Although the same production is found with other parameters

the computational time proves to be less in this case. Also, another sensitivity analysis was made in the case where one day increment is considered to schedule the batches. The optimal parameters for this case are: population size = 90, number of generations = 60, elite rate = 33%, crossover rate = 83% and mutation rate = 1.8% when looking for maximum gas production and: population size = 50, number of generations = 20, elite rate = 25%, crossover rate = 75% and mutation rate = 1% when looking to minimize the computational time.

Also, a design of experiments was made to see which of these parameters are significant and which are not considering both gas production and computational time. The results showed that in the case of computational time the only factor that is considered to be significant is the number of generations. In the case of gas productions that factors that are considered to be significant include the time increment for the residence times, initial population size and number of generations as well as the iterations: time increment*generations, population size*generations, generations*epsilon and population size*crossover.

Chapter 6: Conclusion and Future Work

Biogas is a significant renewable energy resource that helps reduce greenhouse gas emissions such as methane and nitrous dioxide which contribute to global warming. In addition, biogas has the potential to supply 25% of the natural gas demand in the US. Nevertheless, the economic feasibility to produce this gas depends on the ability to manage anaerobic digesters in a cost effective way. A good way to make these systems viable is by maximizing their gas production through the optimization of their feedstocks' scheduling.

Considering that biomass decomposes while in storage, the sequence in which multiple feedstocks are processed in one anaerobic digester system will affect the amount of gas produced. Also, the availability times, biomass quantities and biogas production rates must all be taken into account for maximal biogas production to be achieved during the planning horizon.

This thesis proposed a new Genetic Algorithm to solve the scheduling problem of one anaerobic digester when multiple feedstocks arrive at different times in order to maximize its total gas production. Using the methodology presented, a new optimal solution was found with higher gas production than the Branch and Bound Algorithm with dynamic programming proposed by Gim et al., (2001). In addition, the number of batches can be increased or decreased in the program developed as well as the biogas conversion factors, decay rates, arrival times, total process time and the time increments for residence times to solve many different cases. Furthermore, the parameters to run the Genetic Algorithm can also be changed to increase or decrease the search space for the case being solved.

Moreover, it was demonstrated that Genetic Algorithms can be applied in the production process of biogas effectively, generating optimal scheduling solutions that maximizes gas productivity. In addition, by performing a Design of Experiments through a fractional factorial design, it was concluded which GA's parameters have a significant impact on the optimal solution and the amount of time it takes to find it. Furthermore, it was demonstrated that the GA's parameters can also be optimized to obtain better quality solutions.

The proposed model offered the best solution for the problem of scheduling multiple feedstocks into a single anaerobic digester with a fixed capacity. For future research, the solutions presented can be compared to other optimization algorithms that can also be applied to solve this scheduling problem. Furthermore, this work can be expanded to explore other objective functions, for instance, the integration of more anaerobic digesters with different capacity levels.

References

Abido, M.A. (2002). Optimal design of power-system stabilizers using particle swarm optimization. *IEEE Transactions on Energy Conversion*, 17:406 – 413.

Abu Qdais, H. Bani Han and K. Shatnawi, N. (2010). Modeling and optimization of biogas production from a waste digester using artificial neural network and genetic algorithm. *Resources, Conservation and Recycling* 54, 359–363.

Afshar, M. H. (2007). Partially constrained ant colony optimization algorithm for the solution of constrained optimization problems: Application to storm water network design. *Advances in Water Advances in Water Resources*, Volume 30, Issue 4, p. 954-965.

Aghaie, M., Nazari, T., Zolfaghari , A., Minuchehr,A. and Shirani, A. (2013). Investigation of PWR core optimization using harmony search algorithms. *Annals of Nuclear Energy* 57, 1–15.

Balmant, W. Oliveira, B.H. Mitchell, D.A. Vargas, J.V.C. and Ordonez, J.C. (2014). Optimal operating conditions for maximum biogas production in anaerobic bioreactors. *Applied Thermal Engineering*, Vol. 62 Pages 197-206.

Baojiang, Z. and Shiyong, L. (2007). Ant colony optimization algorithm and its application to Neuro-Fuzzy controller design. *Journal of Systems Engineering and Electronics*, Vol. 18, No. 3, pp.603–610.

Bojic, I. Podobnik, V. Ljubi, I. Jezic, G. and Kusek, M. (2012). A self-optimizing mobile network: Auto-tuning the network with firefly-synchronized agents. *Information Sciences*, Vol. 182 PP. 77–92

Bramley, J. Cheng-Hao Shih, J. Fobi, L. Teferra, A. Peterson, C. Yuan Wang, R. and Rainville, L (2011). Agricultural Biogas in the United States: A Market Assessment. *UEP Field Project Team #6*

Chaudhary, D. (2002). Bee-Inspired Routing Protocols for Mobile Ad HOC Network (MANET). *Journal of Emerging Technologies in Web Intelligence*, vol. 2, pp. 86-88.

Cobb, H. G. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 523-530.

Cohoon, J. P. Martin, W. N. and Richards, D. S. (1991). A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes. *Proc. 4th Int. Conf. Genetic Algorithms*, pp. 244-248.

Colorni A., Dorigo M. and Maniezzo V. (1991). Distributed Optimization by Ant Colonies. *Proceedings of ECAL91 - European Conference on Artificial Life*, Paris, France, Elsevier Publishing, 134–142.

Curry, G. L. and Deuermeyer, B. L. (1986). Optimal scheduling of multiple feedstock batch biogas production systems. *IIE Transactions*, 18, pp. 367–373.

Deuermeyer, B. L. Lee, H. and Curry, G. L. (1986). Optimal residence times for a batch biomass-to-methane conversion systems. *Management Science*, Volume 32. Issue 9.

Ding, Q. Hu, X. Sun, L. and Wang, Y. (2012). An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing*, Vol. 98 pp. 101–107.

Eberhart, R.C. and Hu, X. (1999). Human tremor analysis using particle swarm optimization. *Evolutionary Computation*, 1999. CEC 99. Proceedings of the 1999 Congress.

Fakharudin, A. S. Sulaiman, M. N. Salihon, J. and Zainol N. (2003). Implementing artificial neural networks and genetic algorithms to solve modeling and optimization of biogas production. *Proceedings of the 4th International Conference on Computing and Informatics*, ICOCI Paper No. 088.

Fang, H. L. Ross, P. and Come, D. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling, & open-shop scheduling problems. *Proc. 5th Int. Conf Genetic Algorithms,* 1993, pp. 375-382.

Feldman, R. M. Deuermeyer, B. L. and Curry, G. L. (1987). A dynamic programming optimization procedure for a two-product biomass-to-methane conversion system. *Journal of Mathematical Analysis and Applications*, Volume 125, Issue 1, July 1987, Pages 203–212.

Forster, P. Ramaswamy, V. P. Artaxo, P. Berntsen, T. Betts, R. Fahey, D.W. Haywood, J. Lean, J. Lowe, D.C. Myhre, G. Nganga, J. Prinn, R. Raga, G. Schulz, M. and Van Dorland, R. (2007): Changes in Atmospheric Constituents and in Radiative Forcing. *In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change* [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M.Tignor and H.L. Miller (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Galvez, A. Iglesias, A. and Puig-Pey, J. (2012). Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction. *Information Science,s* 182, 56–76.

Geem, Z. W., and Lee, K, S. (2004). A new structural optimization method based on the harmony search algorithm. *Computers and Structures* 82, 781–798.

Geem, Z. W. Tseng, C. L. and Park, Y. (2005). Harmony search for generalized orienteering problem: best touring in China. *Lect. Notes Comput. Sci,* 3612, 741–750.

Geem, Z.W. (2005). School bus routing using harmony search. *Genetic and Evolutionary Computation Conference*, Washington DC.

Geem, Z.W. (2006). Optimal cost design of water distribution networks using harmony search *Eng.Optim*, Vol. 38, 259–280.

Gim, B. Deuermeyer, B. L. and Curry, G. L. (2001). Optimal scheduling of a single anaerobic digester with multiple feedstocks. *Computers & Industrial Engineering*, Volume 41, Issue 1, Pages 1–15.

Goh, S. J. Gu, D. W. and Man, K. F. (1996). Multi-layer genetic algorithms in robust control system design. *Control '96,* U.K.

Grimaccia, F. Mussetta, M. and Zich, R. (2007). Genetical swarm optimization: Self-adaptive hybrid evolutionary algorithm for electromagnetics. *IEEE Transactions on Antennas and Propagation*, Vol. 55, Issue: 3.

Gueguim Kana, E. B. Oloke, J.K. Lateef, A. and Adesiyan, M. O. (2012) Modeling and optimization of biogas production on saw dust and other co-substrates using Artificial Neural network and Genetic Algorithm. *Renewable Energy,* Vol. 46, pages 276-281.

He, S. Wu, Q.H. Wen, J.Y. Saunders, J.R. and Paton, R.C. (2004) A particle swarm optimizer with passive congregation. *BioSystems*, Vol. 78, Issue 1-3, pages 135–147.

Homaifar, A. and McCormick, E. (1995) Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, *IEEE Trans. Fuzzy Syst.,* Vol. 3, pages 129-139.

IPCC, (2007): Summary for Policymakers. In: *Climate Change 2007: The Physical Science Basis. Contribution of WorkingGroup I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change* [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M.Tignor and H.L. Miller (eds.)]. *Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.*

Kaur, A. and Goyal, S. (2011) A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization. *International Journal of Advanced Science and Technology*, Vol. 29, pages 17-30.

Le Treut, H. Somerville, R. Cubasch, U. Ding, Y. Mauritzen, C. Mokssit, A. Peterson, T. and Prather, M. (2007): Historical Overview of Climate Change. In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the

Intergovernmental Panel on Climate Change [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M. Tignor and H.L. Miller (eds.)]. *Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA*.

Lee, K.S., Geem, Z. W. and Lee, S. H. (2005) The harmony search heuristic for discrete structural optimization. *Eng.Optim*, Vol. 37, pages 663–684.

Lee, Kang S. Geem, and Zong W. (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice" *Computer Methods in Applied Mechanics and Engineering*, Volume 194, Issues 36–38, pages 3902–3933.

Lu, B. Özbakır, L. and Tapkan, P. (2004). Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem. *Computer and Information Science*, Vol. 5, pages 113–144.

Lu, Z. Fan, B. Wang, D. and He, X. (2006). Neural network predictive control based on particle swarm optimization for urban expressway. *Intelligent Control and Automation*, WCICA 2006. The Sixth World Congress on, pages 8606 – 8611.

Mackle, G. Savic, D.A. and Walters, G. A. (1995). Application of genetic algorithms to pump scheduling for water supply. *1st IEE/IEEE Int. Conf on GA 's in Engineering Systems: Innovations and Applications,* Sheffield, U.K,, 1995, pp. 400-405.

Mahdavi, M. Fesanghary, M. and Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, Volume 188, Issue 2, pages 1567–1579.

McCall, J. (2005). Genetic algorithms for modelling and optimization. *Journal of Computational and Applied Mathematics* Vol. 184, pages 205–222.

Mohammadi, S. Mozafari, B. Solimani, S. and Niknam, T. (2013). An Adaptive Modified Firefly Optimisation Algorithm based on Hong's Point Estimate Method to optimal operation management in a microgrid with consideration of uncertainties. *Energy,* Vol. 51 pages 339-348.

Mohandes, M. A. (2012). Modeling global solar radiation using Particle Swarm Optimization (PSO). *Solar Energy* Vol. 86, pages 3137–3145.

Moin, N.H. Zinober, A.S.I. and Harley, P.G. (1995). Sliding mode control design using genetic algorithms. *1st IEENEEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications, Sheffield, U.K*., pages. 238-244.

Nambiar, R. and Mars, P. (1995). Adaptive IIR filtering using natural algorithms. *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, pages. 2011-20110.

National Research Council of the National Academies (2012). Climate Change: Evidence, Impacts, and Choices.

Navrat, P. Jelinek, T. and Jastrzembska, L. (2009). Bee hive at work: A problem solving, optimizing mechanism. *World Congress on Nature & Biologically Inspired Computing*, NaBIC, Coimbatore, pages 122 – 127.

Nhicolaievna, P. N. and Thanh, L. V. (2008). Bee Colony Algorithm for the Multidimensional Knapsack Problem. *Proceedings of the International Multi Conference of Engineers and Computer Scientists Hong Kong,* Vol. 1, pages 20-25.

Olamaie, J. and Niknam, T. (2006). Daily volt/var control in distribution networks with regard to dgs: a comparison of evolutionary methods. *Power India Conference, 2006 IEEE*, page 6.

Oliveira, M. S. Schirru, R. and Medeiros, J. A. C. C. (2009). On the performance of an Artificial Bee Colony Optimization Algorithm Applied to the Accident Diagnosis in a PWR Nuclear Power Plant. *Proceedings of International Nuclear Atlantic Conference - INAC Brazil*, Vol. 27 pages. 1-11.

Omran, M.G.H. and Mehrdad, M. (2008). Global-best harmony search. *Applied Mathematics and Computation* Vol. 198, pages 643–656.

Pacurib, J. A. Seno, G. M. M. and Yusiong, J. P. T. (2009). Solving Sudoku Puzzles Using Improved Artificial Bee Colony Algorithm. *Fourth International Conference on Innovative Computing, Information and Control (ICICIC) Kaohsiung*, pages 885 – 888.

Persson, M. Jonsson, O. and Wellinger, A. (2006). Biogas Upgrading to Vehicle Fuel Standards and Grid Injection. *IEA Bioenergy.*

Pham D.T., Kog E., Ghanbarzadeh A., Otri S., Rahim S. and Zaidi M. (2006) The Bees Algorithm – A Novel Tool for Complex Optimisation Problems, *IPROMS 2006 Proceeding 2nd International Virtual Conference on Intelligent Production Machines and Systems*, Oxford, Elsevier.

Poursalehi, N. Zolfaghari, A. and Minuchehr, A. (2003). Multi-objective loading pattern enhancement of PWR based on the Discrete Firefly Algorithm. *Annals of Nuclear Energy* Vol. 57 pages 151–163.

Ratnaweera, A. Watson, H.C. and Halgamuge, S. K. (2003). Optimization of valve timing events of internal combustion engines with particle swarm optimization. *Evolutionary Computation*, Vol. 4, pages 2411 – 2418.

Robinson, J. Sinton, S. and Rahmat-Samii, Y. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. *Antennas and Propagation Society International Symposium*, IEEE Vol. 1, pages 314 – 317.

Sayadi, M. K. Hafezalkotob, A. and Naini, S. G J. (2013). Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation. *Journal of Manufacturing Systems,* Vol. 32 pages. 78– 84.

Shyu, S. J. Lin, B.M.T. and Yin, P.Y. (2004). Application of ant colony optimization for no-wait flow shop scheduling problem to minimize the total completion time. *Computers & Industrial Engineering* Vol. 47 pages 181–193.

Stanarevic, N. Tuba, M. and Bacanin, N. (2011). Modified Artificial Bee Colony algorithm for Constrained Problems Optimization. *International Journal of Mathematical Models and Methods in applied Sciences*, Vol. 5, pages 644-651.

Teo, C. H. Foo, Y. C. Chien, S. F. Low, A.L.Y. Venkatesh, B. and You.A. H. (2004). Optimal placement of wavelength converters in WDM networks using particle swarm optimizer. *IEEE International Conference on Communications*, pages 1669 – 1673.

Teodorovic, D. and Dell'orco, M. (2005) Bee Colony Optimization – A Cooperative learning approach to Complex Transportation Problems. *Proceedings of the 16th Mini-EURO Conference on Advanced OR and AI Methods in Transportation*, pages 51-60.

Tian, L. and Collins, C. (2004). An effective robot trajectory planning method using a genetic algorithm. *Mechatronics,* Volume 14, Issue 5, Pages 455–470.

United States Environmental Protection Agency (2010). U.S. Anaerobic Digester Status Report.

United States Environmental Protection Agency (2013). Climate Change.

University of New Hampshire AN INTRODUCTION TO THE GLOBAL CARBON CYCLE.

Walthall, C.L., J. Hatfield, P. Backlund, L. Lengnick, E. Marshall, M. Walsh, S. Adkins, M. Aillery, E.A. Ainsworth,C. Ammann, C.J. Anderson, I. Bartomeus, L.H. Baumgard, F. Booker, B. Bradley, D.M. Blumenthal, J. Bunce, K. Burkey, S.M. Dabney, J.A. Delgado, J. Dukes, A. Funk, K. Garrett, M. Glenn, D.A. Grantz, D. Goodrich, S. Hu, R.C. Izaurralde, R.A.C. Jones, S-H. Kim, A.D.B. Leaky, K. Lewers, T.L. Mader, A. McClung, J. Morgan, D.J. Muth, M. Nearing, D.M. Oosterhuis, D. Ort, C. Parmesan, W.T. Pettigrew, W. Polley, R. Rader, C. Rice, M. Rivington, E. Rosskopf, W.A. Salas, L.E. Sollenberger, R. Srygley, C. Stöckle, E.S. Takle, D. Timlin, J.W. White, R. Winfree, L. Wright-Morton, and L.H. Ziska. (2012). Climate Change and Agriculture in the United States: Effects and Adaptation. *USDA Technical Bulletin 1935*, Washington, DC. 186 pages.

Wilson, P. B. and Macleod, M.D. (1993). Low implementation cost IIR digital filter design using genetic algorithms. *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, pages 4-1, 4-8.

Wong, L. P. Low, M. Y. H. and Chong, C. S. (2010). Bee Colony Optimization with Local Search for Traveling Salesman Problem. *International Journal on Artificial Intelligence Tools (IJAIT),* Vol. 19, pages 305-334.

Wu, Z. Zhao, N. Ren, G. and Quan, T. (2009). Population declining ant colony optimization algorithm and its applications. *Expert Systems with Applications,* Vol. 36 pages 6276–6281.

Yan, H. and Ma, R. (2006). Design a novel neural network clustering algorithm based on pso and application. In Intelligent Control and Automation. *WCICA 2006. The Sixth World Congress*, pages 6015 – 6018.

Yang, X. S. (2005). Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach,* pages 317-323.

Yang, X. S. Hosseini, S. S. S. and Gandomi, A. H. (2012). Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect. *Applied Soft Computing*, Vol. 12 pages 1180–1186.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization in: Stochastic Algorithms: Foundations and Applications. *Lecture Notes in Computer Sciences*, Vol. 5792, pages 169-178.

Yang, X. S. (2010). Firefly Algorithm, Stochastic Test Functions and Design Optimisation. *Int. J. Bio-Inspired Computation*, Vol. 2, No. 2, pages 78–84.

Yang, X.-S. (2009). Harmony Search as a Metaheuristic Algorithm in: Music-Inspired Harmony Search Algorithm: Theory and Applications (Editor Z. W. Geem). *Studies in Computational Intelligence.* Springer Berlin, Vol. 191, pages 1-14.

Zecchin, A. C. Simpson, A.R. Maier, H.R. Leonard, M. Roberts, A. J. and Berrisford, M. J. (2006). Application of two ant colony optimisation algorithms to water distribution system optimization. *Mathematical and Computer Modelling,* Vol. 44, pages 451–468.

Zhi, X.H. Xing, X.L. Wang, Q.X. Zhang, L.H. Yang, X.W. Zhou, C.G. and Liang. Y.C. (2004). A discrete pso method for generalized tsp problem. *Machine Learning and Cybernetics, Proceedings of 2004 International Conference*, Vol. 4, pages 2378 – 2383.

Appendix A: Matlab® Code

GA

```
%%%%%%%%%%%%%%%%%%%%%%%%GA for gas production problems%%%%%%%%%%%%%%%%%%%%%%%

%Clear everything
clc;
clear;
%Load Data (data for the problem)
load alpha2.mat
load arrival2.mat
load batches2.mat
load beta2.mat
load ye2.mat
%Define other problem parameters
T=130;       %total period of time
d=1;         %setup time
MTB=5;       %time interval
%Define GA parameters
p_size=100;  %population size
gen=150;     %number of generations
epsilon=.1; %epsilon to end the algorithm
elit=.25;    %elitism prob
crs=.75;     %crossover prob
mut=.01;     %mutation prob

%Start of the algorithm
    % initial population
    [pop]=p_gent( batches,p_size,arrival,T,MTB);
    %Eval initila population
    [gas_prod]=eval_seq2( pop,d,alpha,beta,ye,arrival);
    %initialize conditions to stop GA
    condition1=0;             %generation condition
    currentgeneration=1;      %define current generation
    condition2=0;             %diferrence condition
    currentmax=max(gas_prod); %current max gas production
    progress=max(gas_prod);
    %start while loop
    while condition1+condition2==0
        %reproduction generate elite parents and posible children
        [eliteparents,children] = reproduction( pop,gas_prod,elit,crs,mut );
        % we need to check children consistency
        children=check( children, batches, arrival,T,MTB );
        %create new population
        pop=[eliteparents;children];
        %ensure that the size of the pop = to pop_size
        if size(pop,1)<p_size
            additional=p_gent( batches,p_size-size(pop,1),arrival,T,MTB);
            pop=[pop;additional];
        end
        %eval gas production for new pop
        [gas_prod]=eval_seq2( pop,d,alpha,beta,ye,arrival);
        %update generation
        currentgeneration=currentgeneration+1;
        %check condition 1
```

```
        if currentgeneration==gen
            condition1=1;
        end
        progress=[progress;max(gas_prod)];
        plot(progress);
        [a,b]=max(gas_prod);
        Solution=[pop(b,:),a]
    end
```

```
   P_Gent
 function [pop]=p_gent( batches,p_size,arrival,T,MTB)
%get size #of feedstocks and batches for each feedstock
[a]=size(batches,1);
%get string to choose from
len_chrom=sum(batches);
st=zeros(len_chrom,1);
index=1;
for i=1:a
    for j=1:batches(i)
        st(index)=i;
        index=index+1;
    end
end
st(1,:)=[];
%preallocating pop
pop=zeros(p_size,len_chrom*2);
%generate sequence
for i=1:p_size
    sequence=[1,st(randperm(len_chrom-1))'];
%obtain t times for sequence
tes=zeros(1,len_chrom);
posibles_t=MTB:MTB:T;
indexes_of_tes=ceil(size(posibles_t,2)*rand(1,size(sequence,2)));
tes=posibles_t(indexes_of_tes);
pop(i,:)=[sequence,tes];
end
%checking solutions
pop=check( pop, batches, arrival,T,MTB );
```

Eval_Seq

```
function  Table=eval_seq( seq,MTB,MTC,T,d,alpha,beta,ye,arrival,batches)
%obtain ui times
    ui_times=0:MTB:T;
%Obtain Yi
    for i=1:length(seq)
        i
        %obtain start point for ustates
        if i~=1
            if arrival(seq(i))>= arrival(seq(i-1))
                startt=arrival(seq(i));
                ui=startt:MTB:T;
            else
                startt=arrival(seq(i-1));
                ui=startt:MTB:T;
            end
        else
            ui=arrival(seq(i)):MTB:T;
```

```
        end

        Y_u=[];
        xi=[];
        switch (i)
            case 1
                for index=1:length(ui)
                    Y_u(index)=h(d,seq(i),ye)*g(ui(index)-
arrival(seq(i)),seq(i),alpha,beta,d);
                    xi=ui;
                end
            case length(seq)
                'entro ultimo'
                ui
                %obtaining Yk-1 table from Table
                yukmenos1=cell2mat(Table(1,i-1))
                %evaluate for each uistate
                for index=1:length(ui)
                    %initialize YU_parcial
                    YU_parcial=[]
                    %generare the posible number of xi for specific ui
                    %state
                    xi0=0:MTB:ui(index)-arrival(seq(i))
                    %eval Yu for all the posibles xi
                    for index2=1:length(xi0)

                        'este'
                        xi0(index2)
                        xi0

                        %obtain Yk-1
                        donde=find(yukmenos1(:,1)==(T-xi0(index2)));
                         ui
                        yk=yukmenos1(donde,2)
                        if isempty(yk)==0
                        YU_parcial(index2)=h(T-xi0(index2)-
arrival(seq(i))+d,seq(i),ye)*g(xi0(index2),seq(i),alpha,beta,d)+yk;
                        end
                    end
                    %getting best xi and yu
                    [a,b]=max(YU_parcial);
                    yumax=a;
                    ximax=xi0(b);
                    Y_u(index)=yumax;
                    xi(index)=ximax;
                end
            otherwise
                %obtaining Yk-1 table from Table
                yukmenos1=cell2mat(Table(1,i-1));
                %evaluate for each uistate
                for index=1:length(ui)
                    %initialize YU_parcial
                    YU_parcial=[];
                    %generare the posible number of xi for specific ui
                    %state
                    xi0=0:MTB:ui(index)-arrival(seq(i));
                    %eval Yu for all the posibles xi
                    for index2=1:length(xi0)
```

```
                        %obtain Yk-1
                        donde=find(yukmenos1(:,1)==(ui(index)-xi0(index2)));
                        yk=yukmenos1(donde,2);
                        YU_parcial(index2)=h(ui(index)-xi0(index2)-
arrival(seq(i))+d,seq(i),ye)*g(xi0(index2),seq(i),alpha,beta,d)+yk;
                    end
                    %getting best xi and yu
                    [a,b]=max(YU_parcial);
                    yumax=a;
                    ximax=xi0(b);
                    Y_u(index)=yumax;
                    xi(index)=ximax;
                end
        end

    Table(1,i)={[ui',Y_u',xi']};
    end



end

function valor=h(s,feedstock,ye)
valor=exp(-ye(feedstock)*s);
end

function valor=g(t,feedstock,alpha,beta,d)
if (t-d)<0

    valormax=0;
else
    valormax=t-d;
end
valor=alpha(feedstock)*(1-exp(-beta(feedstock)*valormax));
end
```

Eval_Seq2

```
function  [gas_prod]=eval_seq2( pop,d,alpha,beta,ye,arrival)

%initialize parameters
[a,b]=size(pop);
gas_prod=zeros(a,1);
%obtain seq and tes from chromosoma

for i=1:a
    prod0=0;
    for j=1:b/2
        switch j
            case 1
                y0=h(d,pop(i,j),ye)*g(pop(i,(b/2)+j),pop(i,j),alpha,beta,d);

                prod0=prod0+y0;

            otherwise
```

```
                u=sum(pop(i,(b/2)+1:(b/2)+j));
                x=pop(i,(b/2)+j);
                r=arrival(pop(i,j));
                y0=h(u-x+d-r,pop(i,j),ye)*g(pop(i,(b/2)+j),pop(i,j),alpha,beta,d);
                prod0=prod0+y0;

        end
    end
    gas_prod(i,:)=prod0;
end

end

function valor=h(s,feedstock,ye)
valor=exp(-ye(feedstock)*s);
end

function valor=g(t,feedstock,alpha,beta,d)
if (t-d)<0

    valormax=0;
else
    valormax=t-d;
end
valor=alpha(feedstock)*(1-exp(-beta(feedstock)*valormax));
end
```

Reproduction

```
 function [eliteparents,children] = reproduction( pop,gas_prod,elit,crs,mut )
%%%%%%%%%%%%%%%%%%%%%%Rreproduction Function%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Obtain elite parents
[a,b]=sort(gas_prod,'descend');
eliteparents=pop(b(1:round(elit*size(pop,1)),:),:);
%define how many children we need
howmanychildren=size(pop,1)-size(eliteparents,1);
%initialize children
children=zeros(round(howmanychildren/4)*4,size(pop,2));
%define condition for the while and index for while
condition=0;
index=1;
while condition==0
    %select parents
    index1=0;
    index2=0;
    while index1==index2
        index1=selection(gas_prod);
        index2=selection(gas_prod);
    end
    parent1=pop(index1,:);
    parent2=pop(index2,:);
    %undergo crossover with crs prob
    if crs>=rand()
       children(index:index+3,:)=crossover(parent1,parent2);
       index=index+4;
    end
```

```
    if index==round(howmanychildren/4)*4+1
        condition=1;
    end
end
%Eliminated repeated children
children=unique(children,'rows');
%mutation
for i=1:size(children,1)
    if rand()<=mut
    children(i,:)=mutation(children(i,:));
    end
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%Other Functions%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%selection function
function [winner]=selection(objectives)
option=1;

switch (option)
    case 1
       %Tournament
       index1=0;
       index2=0;
       while index1==index2
            index1=ceil(size(objectives,1)*rand());
            index2=ceil(size(objectives,1)*rand());
       end

       if objectives(index1)>objectives(index2)
           winner=index1;
       else
           winner=index2;
       end

end

end

%crossover function
function [children]=crossover(parent1,parent2)
%initialize children matrix
children=zeros(4,size(parent1,2));
%separate sequences and times
seq1=parent1(1,1:end/2);
seq2=parent2(1,1:end/2);
times1=parent1(1,end/2+1:end);
times2=parent2(1,end/2+1:end);
%crossver for sequences
seq1(:,1)=[]; % eliminate the first feedstook it is always 1
seq2(:,1)=[]; % eliminate the first feedstook it is always 1
crossoverpoint=ceil((length(seq1)-1)*rand());
chilseq1=[1,seq1(1,1:crossoverpoint),seq2(1,crossoverpoint+1:end)];
chilseq2=[1,seq2(1,1:crossoverpoint),seq1(1,crossoverpoint+1:end)];
chilseq3=[1,seq1(1,crossoverpoint+1:end),seq2(1,1:crossoverpoint)];
chilseq4=[1,seq2(1,crossoverpoint+1:end),seq1(1,1:crossoverpoint)];
%crossover for times
```

```
crossoverpoint=ceil((length(times1)-1)*rand());
chiltimes1=[times1(1,1:crossoverpoint),times2(1,crossoverpoint+1:end)];
chiltimes2=[times2(1,1:crossoverpoint),times1(1,crossoverpoint+1:end)];
chiltimes3=[times1(1,crossoverpoint+1:end),times2(1,1:crossoverpoint)];
chiltimes4=[times2(1,crossoverpoint+1:end),times1(1,1:crossoverpoint)];
%form children matrix
children(1,:)=[chilseq1,chiltimes1];
children(2,:)=[chilseq2,chiltimes2];
children(3,:)=[chilseq3,chiltimes3];
children(4,:)=[chilseq4,chiltimes4];

end

%Mutation function
function [mutchild]=mutation(child)
        %obtain seq and times
    seq=child(1,1:end/2);
    seq(:,1)=[];
    times=child(1,end/2+1:end);
    %decide mutating seq or times
    option=ceil(2*rand());
    if option==1
        %mute seq
        %select two points
        point1=0;
        point2=0;
        while point1==point2
            point1=ceil(size(seq,2)*rand());
            point2=ceil(size(seq,2)*rand());
        end
        tmp=seq(1,point1);
        seq(1,point1)=seq(1,point2);
        seq(1,point2)=tmp;
    else
        %mute times
        %select two points
        point1=0;
        point2=0;
        while point1==point2
            point1=ceil(size(times,2)*rand());
            point2=ceil(size(times,2)*rand());
        end
        tmp=times(1,point1);
        times(1,point1)=times(1,point2);
        times(1,point2)=tmp;
    end
    mutchild=[1,seq,times];


    end
```

Check

```
function Corrected=check( chrom, batches, arrival,T,MTB )
%%%%%%%%%%%%function to check and correct chromosomes%%%%%%%%%%%%%%%%%%%%%
%initialize solution
```

```
Corrected=zeros(size(chrom));
%correct batches for the first feedstook
batches(1,1)=batches(1,1)-1;
%initialize no feasible variable
nofeasible=0;
%start cheching each crhom
for i=1:size(chrom,1)

    %obtener seq
    seq=chrom(i,1:end/2);
    seq(:,1)=[];

    %obtain the number of nuber of each type
    count=zeros(size(batches,1),1);
    for j=1:size(batches,1)
        count(j)=size(find(seq==j),2);
    end

    %check if you have more
    %remove batches that are more than allowed
    for k=1:size(batches,1)
        % check is there are more
        if batches(k)<count(k)
            cuantos_hay_de_mas=count(k)-batches(k);
            for m=1:cuantos_hay_de_mas
                %Seacrch batches not allowed
                donde_esta_el_de_mas=find(seq==k);
                punto=ceil(size(donde_esta_el_de_mas,2)*rand());
                seq(donde_esta_el_de_mas(punto))=0;
            end
        end
    end

    %remove batches that are more than allowed
    %add batches that are required
    for k=1:size(batches,1)
        % check is there are more
        if batches(k)>count(k)
            cuantos_faltan=batches(k)-count(k);
            for m=1:cuantos_faltan
                %Seacrch batches not allowed
                donde_hay_zeros=find(seq==0);
                punto=ceil(size(donde_hay_zeros,2)*rand());
                seq(donde_hay_zeros(punto))=k;
            end
        end
    end



    %complete seq
    seq=[1,seq];

    %Correcting times
    times=chrom(i,end/2+1:end);
    %min_time=MTB;
    for m=1:size(times,2)-1
        suma_tiempos_prev=sum(times(1:m))-times(m);
```

```
        timepo_min_remaining=MTB*(size(times,2)-m);
        max_time=T-suma_tiempos_prev-timepo_min_remaining;
        maximo_de_arrivals=arrival(seq(m+1))- suma_tiempos_prev;
        if MTB<maximo_de_arrivals
            min_time=maximo_de_arrivals;
        else
            min_time=MTB;
        end
        % check if the solution is feasible
        if min_time<=max_time
            %time is feasible continue
            %check if the current time is feasible or not
            if times(m)>=min_time && times(m)<=max_time
                % it is correct do nothing
            else
                %correct the time
                %create posible times values
                posible_times=min_time:MTB:max_time;
                times(m)=posible_times(ceil(size(posible_times,2)*rand()));
            end
        else
            %time ans sequence is not feasible record that
            nofeasible(i)=1
        end
       % pause();
    end
    times(end)=T-sum(times(1:end-1),2);

%     %check if the time is correct
%     check1=checktime(times,T);
%     check2=checkarrivals(seq,times,arrival);
%
%
%     while (check1+check2)>0
%         if check1>0
%           times=correctcheck1(times,T,MTB) ;
%
%         end
%         if check2>0
%           times=correctcheck2(seq,times,arrival);
%
%         end
%         check1=checktime(times,T);
%         check2=checkarrivals(seq,times,arrival);
%
%     end
     Corrected(i,:)=[seq,times];



end
no_feasible_sol=find(nofeasible==1);
Corrected(no_feasible_sol,:)=[];

end
```

Vita

Ana C. Cram was born in Ciudad Juarez, Chihuahua, Mexico. The fourth child of Armando Davila Cardenas and Martha Dolores Mena Olveda. She graduated with a bachelor's degree in Industrial Engineering from the University of Texas at El Paso (UTEP) in spring 2011. Then in spring 2012 she started her Master of Science studies in the Industrial, Manufacturing and Systems Engineering department at UTEP while working with Dr. Jose Espiritu as a Research Assistant.

Permanent address: 10200 Hedgerow Apt. 23
El Paso, TX. 79925

This thesis/dissertation was typed by Ana C. Cram.