UNIVERSITY OF CALGARY

Optimization of Pipeline Operations Using Biologically-Inspired Computational Models

by

Thamar Elena Mora

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

SEPTEMBER, 2008

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Optimization of Pipeline Operations Using Biologically-Inspired Computational Models" submitted by Thamar Elena Mora in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Supervisor, Dr. Abu B. Sesay
Department of Electrical and Computer Engineering

Co-Supervisor, Dr. Joerg Denzinger
Department of Computer Science

Dr. Guenther Ruhe
Department of Electrical and Computer Engineering

Dr. Anil K. Mehrotra
Department of Chemical and Petroleum Engineering

Dr. Rob C. Kremer
Department of Computer Science

Dr. Hossein Golshan
TransCanada PipeLines Limited

External Examiner, Dr. Kay C. Wiese
Simon Fraser University

September 17, 2008
Date

Abstract

This thesis presents a multi-agent cooperative search approach to optimize the operation of large and complex natural gas pipeline networks. The proposed approach is based on a biologically-inspired computational model, namely particle swarm optimization, and the main objective is to determine the control parameters of a natural gas transmission pipeline network that result in optimal operation while maintaining the desired throughput and satisfying given system constraints. Solving this problem is not trivial given the large number of decision variables and constraints, the nature of the objective functions and the limited time available to obtain the solution. The used cooperative search approach improves on the pure competition of search agents by the sporadic exchange of solutions which are integrated into the search state of the search agents and also used to improve their search control. The approach exploits the strength of particle swarm optimization in each agent to deal with high-dimensional problems that include a mix of discrete and continuous decision variables. Incorporation of domain knowledge into the search allowed the identification of solutions that can be pre-evaluated without having to use the time consuming pipeline simulations used

to measure the quality of a solution. The experimental evaluation with real problem instances from TransCanada PipeLines Ltd. show that the developed system meets the industry's time requirements and, for large and complex pipeline networks, it reliably outperforms the current state-of-the-art interactive method by creating solutions that require at least 12% less energy, reducing not only the transportation costs but also the amount of greenhouse gas emissions being dissipated to the atmosphere.

Acknowledgements

I would like to thank my supervisors for their guidance and encouragement; and TransCanada PipeLines Ltd. in particular L. Stein, H. Golshan, G. Poissant, C. Konecnik, D. Miller, B. Huynh, and K. Jungowski for the collaboration and valuable discussion throughout the course of this work.

Dedication

To my parents.

Table of Contents

List of Tables

List of Figures

List of Symbols

	Optimization Theory
MOO	Multi-objective optimization
i, j	Indexes
$\vec{x}$	Vector of decision variables
x_i	Decision variable i
x^c	Continuous decision variable
x^d	Discrete decision variable
n_x	Dimension of the optimization problem
n_c	Number of continuous decision variables
n_d	Number of discrete decision variables
$\vec{f}(\vec{x})$	Vector of objective functions
$f_i(\vec{x})$	Objective function i
n_{obj}	Number of objectives
$\vec{g}(\vec{x})$	Vector of inequality constraints
$g_i(\vec{x})$	Inequality constraint i
n_{ineq}	Number of inequality constraints

$\vec{h}(\vec{x})$	Vector of equality constraints
$h_j(\vec{x})$	Equality constraint j
n_{eq}	Number of equality constraints
$\prec$	Dominance
$\mathcal{P}^*$	Pareto-optimal set
$\mathcal{PF}$	Pareto front
$n_{\mathcal{PF}}$	Maximum number of solutions in $\mathcal{PF}$
	NGPO
GHG	Greenhouse gas emissions
MW	Megawatt
NGPO	Natural gas pipeline operations
$f_1(\vec{x})$	Fuel consumption
$f_2(\vec{x})$	Throughput
$f_3(\vec{x})$	Linepack
$\Omega(\vec{x})$	Violations caused by $\vec{x}$
$\mathcal{HC}(\vec{x})$	Hydraulic correctness of solution vector $\vec{x}$
$\dot{m}$	Mass flow rate
P_s	Suction pressure
P_d	Discharge pressure
CS	Compressor station
n_{CS}	Number of compressor stations in a network

Q_i	Flow at node i
n_{DN}	Number of delivery nodes
C	Lumped constant for linepack calculation
L	Length of the pipe
D	Internal diameter of the pipe
P_a	Average pressure
Z_a	Average compressibility factor
T_a	Average temperature
Sg	Surge limit
St	Stonewall limit
CU	Compressor unit.
Sp^-	Minimum compressor unit speed
Sp^+	Maximum compressor unit speed
HP^-	Minimum power
HP^+	Maximum power
n_{cu_i}	Number of compressor units at CS_i
Ψ_i	Cost of operation of CS_i
D_{ij}	Feasible domain of compressor unit j at CS_i
H	Compressor adiabatic head
Q	Compressor volumetric flow rate
Sp	Compressor speed

η	Adiabatic efficiency
K	Adiabatic gas exponent
Z	Gas compressibility factor
R	Gas constant
T_s	Suction temperature
φ_j	Fuel cost function of compressor unit j
α	Constant
CV	Control valve
BV	Block valve
Pr	Variable associated to volume of gas received
n_{Pr}	Number of segments in which Pr is partitioned
	——— Search Systems ———
s_i	Search state i
$\mathcal{A}$	Search model
$\mathcal{F}$	Set of facts
$\mathcal{E}xt$	Set of extension rules
$\mathcal{P}$	Search process
$\mathcal{E}nv$	Search environment
$\mathcal{K}$	Search control
$\mathcal{S}$	Set of all possible search states
Ins	Search instance

s^{init}	Initial search state
$\mathcal{G}$	Termination condition of the search
SP	Search problem
s^{new}	New search state
$\mathcal{SS}$	Solution space
$\mathcal{SS}^{init}$	Reduced solution space
n_{init}	Number of search steps using $\mathcal{SS}^{init}$
$\mathcal{VS}$	Velocity space
$\mathcal{MASS}$	Multi-agent search system
$\mathcal{A}g_S$	Start agent
$\mathcal{A}g_E$	End agent
$\mathcal{A}g_i$	Search agent i
$n_{\mathcal{MASS}}$	Number of search agents in a $\mathcal{MASS}$
$\mathcal{K}om$	Communication structure
TECHS	TEams for Cooperative Heterogeneous Search
$\mathcal{A}g_M$	Master agent
mes	Communication function
$\mathcal{G}_{comm}$	Communication condition
n_{comm}	Number of search steps between communications
PU	Set of processing units
pu	Processing unit

n_{pu}	Number of processing units available
k_i^+	Positive information to influence $\mathcal{K}$
sol_i^+	Positive information to influence the search state
$n_{k_i^+}$	Frequency to update k_i^+
$n_{sol_i^+}$	Frequency to update sol_i^+
$\mathcal{G}_{comm_{k_i^+}}$	Communication condition for k_i^+ information
$\mathcal{G}_{comm_{sol_i^+}}$	Communication condition for sol_i^+ information
	PSO
PSO	Particle swarm optimization
p_i	Particle i
$\vec{x}_i$	Position of particle i
$\vec{v}_i$	Velocity of particle i
$\vec{x}_i^{new}$	New position of particle i
$\vec{v}_i^{new}$	New velocity of particle i
n_{swarm}	Number of particles in a swarm
$\vec{x}_{BG}$	Best position found by the whole swarm
$\vec{v}_{BG}$	Velocity associated to $\vec{x}_{BG}$
$\vec{x}_{BO_i}$	Best position found so far by particle i
$\mathcal{W}$	Inertia weight
C_1	Cognitive learning factor
C_2	Social learning factor

r	Random value
$\vec{x}^*$	Optimal solution vector
x_i^d	Discrete decision variable i
x_i^c	Continuous decision variable i
n_d	Number of discrete decision variables
n_c	Number of continuous decision variables
$\vec{\mathcal{R}}$	Vector with variability range of decision variables
$\mathcal{R}_i$	Variability range of decision variable i
$\mathcal{M}_{CS}$	Pre-evaluation function
$\vec{0}$	Zero vector
$\lhd$	Comparison operator
$\lhd_{\text{MOO}}$	Comparison operator in MOO
mut	Mutation operator
P_{mut}	Probability of mutation
p^{init}	Initial particle
$\vec{x}^{init}$	Initial position of a particle
$\vec{v}^{init}$	Initial velocity of a particle
$\vec{x}_{BO}^{init}$	Initial best position of a particle
n_{PHASEI}	Number of iterations in PHASE I
n_{PHASEII}	Number of iterations in PHASE II

Epigraph

Positive attract positive.

~ T

1 Introduction

The goal of the research work in this thesis is to optimize natural gas pipeline operations within the time frame required by the transportation industry. To achieve this goal we propose a biologically-inspired computational model to find the set of operational settings that optimizes the operation of natural gas pipeline networks.

A **pipeline transmission system** is represented by a complex network that may consist of hundreds of nodes, devices and other equipment to control. Approximately 95% of Canada's crude oil and natural gas is transported by pipelines. Natural gas is generally received from receipt points along the pipeline network and delivered to sales stations at specified flows and pressures. Between these points a pressure drop occurs due to gas expansion, friction loss, changes in elevation and changes in temperature [Mohitpour et al. 2003]. Compression is required to overcome the pressure losses that occur over the length of the pipeline. Adding compressor stations at intervals along the pipeline network is one of the solutions used to achieve and maintain the required pressure. Just in TransCanada's Alberta Gas Transmission System there are more than 50 compressor stations with at least two compressor units in each station. This quantity neither includes gathering networks nor the facilities from other transportation

companies.

A typical **compressor station** can include one or several compressor units which in turn can be of different type, i.e. different models with different capacity and efficiency. Natural gas-fueled turbine engines are the most common drivers for compressors on natural gas transmission pipelines in Canada. Gas turbines spin centrifugal compressors and compress the gas up to a hundred times atmospheric pressure to move the gas. The amount of gas used by turbines can reach 3-5% of the transported gas [Wu 1998]. According to [on Trade and UNCTAD 2004] North America's natural gas consumption in 2003 was 27.5 Tcf . Using 2003's industrial gas prices of $5.78 USD/Mcf [of America IPAA], an average of 4% of transportation costs (1.1 Tcf) would represent $6.35 billion USD. A small improvement in pipeline operations could lead to substantial savings in operating costs; for the 2003 example an improvement of 1% could save $63.5 million USD.

Reduction of the energy used in pipeline operations not only has a tremendous economical impact but also quite an environmental one. More efficient operation of compressor stations results in less greenhouse gas (GHG) emissions being dissipated to the atmosphere. More than 50% of the total human-caused GHG emissions result from the production and use of energy [Foundation 1999]. About 70% of GHG emissions from natural gas occur when natural gas is burned to produce heat or energy [Foundation 1999]. Pipelines emit carbon dioxide (CO_2) mainly due to energy used at the compressor stations. Combustion of natural gas generates mostly CO_2 and water

vapor, the same substances emitted when people breathe. For each megawatt (MW) of energy produced (35% th. eff.) five tonnes of CO_2 are generated per year [Botros et al. 2004]; i.e. a transportation company with capacity of 3,000 MW would emit 15,000 tonnes of CO_2 per year. CO_2 is one of the gases that contribute to the greenhouse effect; the others are methane (CH_4) and nitrous oxide (NO_2). Pipeline companies reduce GHG emissions mainly by improving the use of energy acquiring more efficient equipment and by adopting better operating practices.

Operation of a natural gas pipeline network implies the selection of all operational settings of the components of the network in order to maintain not only the desired throughput but also to meet the standards and regulations designed to minimize the risk of high-pressure transmission lines. It turns out that for specific throughput requirements there are several options to operate the compressor units at the compressor stations and achieve these requirements. Each of these options commonly implies different transportation costs.

The research in this thesis addresses the problem of determining the operational configuration of compressor stations and other pipeline devices that uses the minimum amount of energy (e.g. fuel, power) for given transportation requirements. The configuration includes, for example, the set of compressors that should be ONLINE/OFFLINE, and if ONLINE, the corresponding level of operation. The solution procedure should lead to the result in a short period of time to enable optimization of operations as often as necessary to keep the process as close as possible to optimal conditions.

To tackle this problem we propose a multi-agent cooperative search system in which the search agents explore the search space and exchange solutions from time to time to update the search state and improve the search control. We use particle swarm optimization as set-based search and test the proposed approach using real problem instances of TransCanada PipeLines Ltd. Close interaction with experts in pipeline operations allowed the identification of domain knowledge useful to improve the efficiency of the search. This knowledge was incorporated and used to identify solutions that can be evaluated without using the time-intensive pipeline simulations that are normally needed to measure the quality of a solution.

Results of the experiments show that the proposed multi-agent cooperative search system can deal with high-dimensional problems that include a mix of discrete and continuous decision variables. Results demonstrate that the proposed approach meets TransCanada's time requirements and confirms that, for large and complex pipeline networks our approach reliably outperforms the current state-of-the-art interactive method used by industry by creating solutions that require at least 12% less energy. In addition, we present the extension of the multi-agent cooperative search system to the multi-objective optimization case, again proving that the proposed approach is capable of solving complex pipeline sub-networks in a timely manner.

This thesis is organized as follows. Chapter 2 introduces preliminary concepts of optimization theory followed by the description of the main components of a pipeline sub-network that can be optimized to reduce the cost of transportation and GHG emis-

sions. Chapter 3 introduces fundamental concepts of search systems, particle swarm optimization, as well as distributed search systems. Our proposed approach to optimize natural gas pipeline operations is introduced in Chapter 4. Chapter 5 presents the extension of our approach to the multi-objective optimization case. Chapter 6 summarizes related work relevant to the various topics covered in this thesis. Chapter 7 presents the results of the experiments set to analyze the performance of our approach and compare it against other methods. And Chapter 8 summarizes the work done during this research and provides directions for future work.

2 Optimization of Natural Gas Pipeline Operations - Basic Concepts

The first part of this chapter is an introduction to preliminary concepts of optimization theory for single and multi-objective problems. The second part explains the main components that can be optimized while operating a natural gas transportation system including some fundamentals of hydraulics. The chapter concludes with a description of the optimization of pipeline operations as a search problem.

2.1 Preliminary Concepts of Optimization

Before dealing explicitly with the problem of optimization of natural gas pipeline operations we describe a general multi-objective optimization problem as:

$$Minimize\ \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_{n_{obj}}(\vec{x})] \tag{2.1}$$

subject to

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, ..., n_{ineq} \tag{2.2}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., n_{eq} \tag{2.3}$$

where $\vec{x} = (x_1, x_2, ..., x_{n_x})$ is the vector of decision variables, $f_i(\vec{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}, i = 1, 2, ..., n_{obj}$ are the objective functions and $g_i(\vec{x}), h_j(\vec{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$, $i = 1, 2, ..., n_{ineq}$, $j = 1, 2, ..., n_{eq}$ are constraint functions of the problem, n_{ineq} and $n_{eq} \in \mathbb{N}$.

Most real-life optimization problems are multi-objective. For single-objective optimization problems the optimum is the best of a set of candidate solutions. But for multi-objective optimization (MOO) the definition of an optimum is not that simple. This is because of the potentially conflicting objectives: while some solutions may favor an objective they may deteriorate other objectives at the same time.

There are different options to deal with MOO: aggregation, ordering, and Pareto dominance. Aggregation methods aim for the optimization of a weighted sum of the objectives and defining the weight of each objective is problem dependent. Ordering methods consider one objective function at a time based on a specific criterion, e.g lexicographic ordering of the objectives which might be sensitive to the order assigned to the objectives. Pareto dominance establishes the notion of preference to decide if one solution is better than other considering the value of all the objectives. **Pareto dominance** is used in this research work and the main concepts are provided below.

The set of solutions that cannot improve any objective without degrading one or several objectives is called the set of non-dominated solutions. This set is also called the **Pareto-optimal set** and the corresponding set of vectors of objective functions is called **Pareto front**. The concept of Pareto-optimality was first introduced by Francis Ysidro Edgeworth but later conceptualized by the Italian economist Vilfredo Pareto in

his work *Manual of Political Economy* in 1906 [Engelbrecht 2005].

The following definitions are from [Reyes-Sierra and Coello 2006].

Definition 2.1: Given two vectors $\vec{f_1}$ and $\vec{f_2} \in \mathbb{R}^{n_{obj}}$, we say that $\vec{f_1} \leq \vec{f_2}$ if $\vec{f}_{1,i} \leq \vec{f}_{2,i}$ $\forall i = 1, \ldots, n_{obj}$, and that $\vec{f_1}$ **dominates** $\vec{f_2}$ (denoted by $\vec{f_1} \prec \vec{f_2}$) if $\vec{f_1} \leq \vec{f_2}$ and $\vec{f_1} \neq \vec{f_2}$.

Definition 2.2: A vector of decision variables $\vec{x} \in X \subset \mathbb{R}^{n_x}$ is **nondominated** with respect to X, if there does not exist another $\vec{x}' \in X$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Definition 2.3: A vector of decision variables $\vec{x}^* \in F \subset \mathbb{R}^{n_x}$ is **Pareto-optimal** if it is nondominated with respect to F, where F is the feasible region.

Definition 2.4: The **Pareto-optimal set** ($\mathcal{P}^*$) is defined by $\mathcal{P}^* = \{\vec{x} \in F \mid \vec{x}$ is Pareto-optimal$\}$.

Definition 2.5: The **Pareto front** ($\mathcal{PF}^*$) is defined by $\mathcal{PF}^* = \{\vec{f}(\vec{x}) \in \mathbb{R}^{n_{obj}} \mid \vec{x} \in \mathcal{P}^*\}$.

One goal of MOO algorithms is to minimize the distance between the set of non-dominated solutions obtained by the algorithm and the true Pareto front. For some problems the true Pareto front is not known and as a consequence this goal cannot be pursued. Another important goal is to promote diversity of the solutions by using operators to attract decision vectors towards less populated areas of the Pareto front. A third goal is aimed to maintain a record of the non-dominated solutions already found, a kind of elitist strategy used by maintaining a repository of these solutions.

$\mathcal{P}^*$ is the set of solutions from where decision makers can select a solution. The maximum number of accepted solutions in a Pareto-optimal set is represented by $n_{\mathcal{PF}}$.

Restrictions on the size of the archive have an impact on the diversity of solutions and the computation time. There must be a balance between the requirements of a satisfactory representation of $\mathcal{P}^*$ and $\mathcal{PF}^*$ and the time required to execute the MOO search algorithm.

The concepts introduced above and the definitions of Pareto dominance are necessary for the comparison of solutions in the MOO problem which will be introduced in Chapter 5.

2.2 Optimization of Natural Gas Pipeline Operations

Operation of a natural gas pipeline network implies the selection of all operational settings of the components of the network in order to maintain not only the desired throughputs but also to meet the standards and regulations designed to limit the inherent risk of high-pressure transmission lines to a minimum.

Optimization of natural gas pipeline operations (NGPO) can be achieved by optimizing objectives such as fuel consumption, throughput and *linepack*[1] [Botros et al. 2004, 2006]. The objective of this research work is, in addition to optimize $\vec{f}(\vec{x})$, also to reduce the computational cost necessary to find the vector of operational settings $(\vec{x})$ that optimizes the operation of pipeline transmission networks, i.e. achieve a minimal (or nearly minimal) cost of transportation of natural gas while satisfying safety regulations and meeting customer demands.

[1]Volume of gas contained in a pipeline system at any point in time.

More formally, the main variables that affect the fuel consumption are the mass flow rate ($\dot{m}$), suction pressure (P_s) and discharge pressure (P_d) at each compressor station (CS). The cost of operation of CS_i can then be expressed as $\Psi_i(\dot{m}, P_s, P_d)$. The objective function $f_1(\vec{x})$ for the fuel minimization problem can then be calculated as:

$$Minimize \quad f_1(\vec{x}) = \sum_{i}^{n_{CS}} \Psi_i(\dot{m}, P_s, P_d) \tag{2.4}$$

where n_{CS} is the total number of CSs in the pipeline network and $\{\dot{m}, P_s, P_d\} \in \vec{x}$.

The maximum throughput can be represented by:

$$Maximize \quad f_2(\vec{x}) = \sum_{i=1}^{n_{DN}} \mathcal{Q}_i \tag{2.5}$$

where $\mathcal{Q}_i$ is the flow at delivery node i, and n_{DN} is the number of delivery nodes.

According to [Botros et al. 2004] the maximum linepack can be formulated by the following objective function[2]:

$$Maximize \quad f_3(\vec{x}) = C\frac{LD^2P_a}{Z_aT_a} \tag{2.6}$$

where C is a lumped constant for linepack calculation, L is the length of the pipe, D is the internal diameter of the pipe, P_a the average pressure, Z_a the average compressibility factor, and T_a the average temperature.

Some of the constraints imposed to the solution of $f_1(\vec{x})$ are governed by the physical characteristics of each compressor unit such as *surge*[3] (Sg) and *stonewall*[4] (St) limits,

[2]For a given segment of pipe.

[3]Limit set to avoid unstable conditions (pulsating flow) in centrifugal compressors operating under low flow conditions.

[4]High flow condition in which the velocity of the fluid can approach sonic speed.

minimum (Sp^-) and maximum (Sp^+) speed, and minimum (HP^-) and maximum (HP^+) power. The set of constraints for each compressor unit j at CS_i is given by:

$$HP^-_{i,j} < HP_{i,j} < HP^+_{i,j} \tag{2.7}$$

$$Sp^-_{i,j} < Sp_{i,j} < Sp^+_{i,j} \tag{2.8}$$

$$Sg_{i,j} < Q_{i,j}/Sp_{i,j} < St_{i,j} \tag{2.9}$$

where $j = 1, ..., n_{cu_i}$, and n_{cu_i} is the total number of compressor units at CS_i.

The feasible domain D_{ij} of compressor unit j at CS_i is determined in terms of adiabatic head (H_j), volumetric flow rate (Q_j), speed (Sp_j), and adiabatic efficiency (η_j). The relationship between H, Q, Sp, and η can be described by [Percell and Ryan 1987]:

$$H/Sp^2 = h_1 + h_2(Q/Sp) + h_3(Q/Sp)^2 + h_4(Q/Sp)^3 \tag{2.10}$$

$$\eta = e_1 + e_2(Q/Sp) + e_3(Q/Sp)^2 + e_4(Q/Sp)^3 \tag{2.11}$$

where h_r and e_r $(r = 1, 2, 3, 4)$ are constants that depend on the compressor unit j. H, Q and S are related to $\dot{m}$, P_s and P_d by:

$$H = ZRT_s \frac{K}{K-1} \left[\left(\frac{P_d}{P_s} \right)^{\frac{K-1}{K}} - 1 \right] \tag{2.12}$$

$$Q = ZRT_s \frac{\dot{m}}{P_s} \tag{2.13}$$

where Z is the gas compressibility factor, R the gas constant, T_s the suction temperature, and K the adiabatic gas exponent. From Equation 2.9, $Q_{i,j}$ may take any value

within the interval $[Q^-_{i,j}, Q^+_{i,j}]$ determined by $Q^-_{i,j} = Sp^-_{i,j} \cdot Sg_{i,j}$ and $Q^+_{i,j} = Sp^+_{i,j} \cdot St_{i,j}$. $H_{i,j}$ is lower bounded by $Sp^-_{i,j}$ and $St_{i,j}$, and upper bounded by $Sp^+_{i,j}$ and $Sg_{i,j}$.

The fuel cost function $\varphi_j(\dot{m}, P_s, P_d)$ for compressor unit j can be calculated as:

$$\varphi_j(\dot{m}, P_s, P_d) = \alpha \frac{\dot{m}H}{\eta} \tag{2.14}$$

where α is constant.

Other sets of constraints of the form of $g(\vec{x})$ and $h(\vec{x})$ as in Eqs. 2.2 and 2.3 are governed by pipeline characteristics such as operating pressure limits, mass flow balance equations and flow equations. Although a detailed description of these equations is out of the scope of this work, a good reference for pipeline hydraulics can be found in [Mohitpour et al. 2003].

In addition to the compressor settings described above, other hydraulic components that belong to $\vec{x}$ and affect $\vec{f}(\vec{x})$ are the settings of all control valves (CV) and block valves (BV) in the transportation system, as well as the availability of gas supply (receipts) and market demand (deliveries). It is worth noting that some of the components of the solution vector $\vec{x}$ are discrete decision variables such as the status of the CSs (ONLINE/OFFLINE) and the status of BVs (FULLYOPENED/FULLYCLOSED), hence adding complexity to the solution surface and as a consequence adding difficulty to the problem.

Why is the optimization of NGPO a difficult problem to solve?

Solution of the optimization of pipeline operations problem is definitely not straightforward due to the non-convexity of D_{ij}, non-linearity and non-convexity of Ψ_{ij}, and non-convexity of the set defined by the pipe flow equations [Wu 1998].

The dimension of the vector $\vec{x}$ ($\|\vec{x}\| = n_x$) of the optimization problem directly depends on the size (n_{CS}) and configuration[5] of the pipeline network under investigation and the number of parameters considered sufficient to define its operation. A typical network might consist of thousands of pipes, dozens of CSs with several compressor units inside, meter stations, cooling systems and a large number of different devices such as CVs and BVs. The complexity of the optimization problem grows with n_x, the well known curse of dimensionality problem.

Solving this problem has proven to be computationally expensive. The big challenge is to identify the set of pipeline operational settings —e.g. pressure at control nodes, mass flow rates, compressors settings, status of block valves, etc.— that optimizes objectives such as the fuel consumption within an acceptable time frame. The solution space of $\vec{f}(\vec{x})$ may be composed by many, from hundreds to millions, of combinations of operational settings that satisfy Equations 2.2 and 2.3 but the goal of this research work is to identify the combination of operational parameters in $\vec{x}$ (or sets of combinations) that optimizes $\vec{f}(\vec{x})$ and to do this in a timely manner.

Optimization is a search for an optimum solution by iteratively transforming a current candidate solution into a new, hopefully better, solution [Engelbrecht 2005]. In

[5]E.g. gun-barrel, serial, parallel, looped, etc.

other words, optimization of NGPO is a search for the vector $\vec{x}$ that optimizes $\vec{f}(\vec{x})$. In this context, Chapter 3 introduces fundamental concepts of search systems and the importance of the design parameters to speedup the search. Then, Chapter 4 presents the proposed search system capable of optimizing, within the given time requirements, the operation of large and complex natural gas pipeline networks while handling all standard and additional non-standard components of large pipeline networks. Chapter 5 presents the extension to the multi-objective optimization case. The proposed approach has been tested using several real world scenarios and the results are presented and discussed in Chapter 7.

3 Search Systems

Optimization of pipeline operations involves continuous and discrete decision variables. As a result, search methods that evaluate more and more instantiated partial solutions such as Branch-and-Bound or A*, are not easily applicable to this type of optimization problem. The fact that the quality of a solution has to be measured using a pipeline simulator completely excludes these approaches since a partial solution will trigger an error message from the pipeline simulator without providing other useful information for the search control. Therefore the search methods used in this research to solve the NGPO problem are instantiations of the so-called set-based search approach in which a search state consists of one or several (a set of) solutions, and the transition operators use all or some of the solutions in the current state to create new solutions. Examples for set-based search are hill-climbing, simulated annealing, tabu search[6], genetic algorithms, evolutionary strategies, and particle swarm systems. We selected particle swarm systems to solve the NGPO problem.

The objective of this chapter is to introduce the basic terminology of set-based search, explore the principles of particle swarm optimization, and conclude with con-

[6]All these use only one solution in a state.

cepts for distributed search systems. In other words, this chapter covers the fundamental principles on which our proposed approach for solving the NGPO problem is based.

3.1. Set-based Search

In general, a search can be described as traversing a sequence of search states. A **search state** (s) describes the progress a search has made and therefore contains many pieces of information. The following terminology of set-based search is taken from [Denzinger 2000]. A set-based **search model** $\mathcal{A}$ is defined by a set of **facts** $\mathcal{F}$ and a set of **extension rules** $\mathcal{E}xt$ that describes all the valid transitions between the states.

A set-based **search process** $\mathcal{P}$ is defined by the triple $\mathcal{P} = (\mathcal{A}, \mathcal{E}nv, \mathcal{K})$, where $\mathcal{E}nv$ is the environment of the search process which models external influences, information not included in a search state, and perhaps also knowledge about the problem to be solved. The **search control** $\mathcal{K}$ guides the search process by selecting the next step of the search based on the current state and the environment, i.e. $\mathcal{K} : S \times \mathcal{E}nv \rightarrow S$, where S is the set of possible search states. The quality of the design of $\mathcal{K}$ has strong impact on the efficiency of the search process, e.g. the computation time. If there is interest on improving a search algorithm, spending some time on the careful design of $\mathcal{K}$ will surely pay off.

A set-based **search instance** Ins is defined by the tuple $Ins = (s^{init}, \mathcal{G})$, where

$s^{init} \in S$ is the initial state of the search, and $\mathcal{G} : S \rightarrow \{\text{TRUE}, \text{FALSE}\}$ is the termination condition of the search. Examples of termination conditions are: terminate when a maximum number of search steps or time limit has been reached, when an acceptable solution has been found, when no improvement is observed over a number of transitions, etc..

Set-based search is an appropriate approach to model and solve search problems characterized by little structure. Examples of this kind of problems are the deductive generation of explicit knowledge that is used in many expert systems, and the solution of optimization problems for which, in addition to the function to optimize, only general knowledge is available to find a good suboptimal solution [Denzinger 1999].

3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based search algorithm inspired by the social behavior of bird flocks in their hunt for food, and was originally proposed in 1995 [Eberhart and Kennedy 1995],[Kennedy and Eberhart 1995]. The main idea is that the behavior of an individual, also called particle, is influenced by its own experience (cognitive learning) and by the experience of successful individuals in its group (the social behavior).

The position of a particle is represented by $\vec{x}_i(t)$. The position is updated by adding the velocity vector $\vec{v}_i(t)$ to the current position as follows:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{3.1}$$

The velocity vector $\vec{v}_i(t)$ is calculated as:

$$\vec{v}_i(t) = \mathcal{W}\vec{v}_i(t-1) + C_1 r_1(\vec{x}_{BO_i} - \vec{x}_i(t-1)) + C_2 r_2(\vec{x}_{BG} - \vec{x}_i(t-1)) \tag{3.2}$$

where $\vec{x}_{BO_i}$ is the particle's best position found so far and $\vec{x}_{BG}$ is the best position found so far by the whole swarm. $\mathcal{W}$ is the inertia weight that controls the effect of the previous velocity, C_1 is the cognitive learning factor that controls the influence of the particle's own experience, C_2 is the social learning factor that controls the influence of the other particles, and $r_1, r_2 \in [0, 1]$ are random values. C_1 and C_2 are usually defined as constants and sometimes are referred to as trust parameters where C_1 expresses how much confidence a particle has in itself and C_2 expresses how much confidence a particle has in its neighbors. The general PSO algorithm is depicted in Table 3.1.

Large values for $\mathcal{W}$ facilitate exploration with increased diversity while small values for $\mathcal{W}$ promote local exploitation. The optimal value for $\mathcal{W}$ is problem-dependent [Shi and Eberhart 1998]. There have been theoretical studies about the convergence properties of PSO which have concluded that the performance of PSO is sensitive to the selection of the control parameters $\mathcal{W}$, C_1, and C_2 [Engelbrecht 2005]. Studies in [Ozcan and Mohan 1998] and [Ozcan and Mohan 1999] concluded that, when $0 < \phi < 4$, where $\phi = \phi_1 + \phi_2$, and $\phi_i = C_i r_i, i = 1, 2$, the trajectory of a particle is a sinusoidal wave where the initial conditions determine the amplitude and frequency of the wave. This periodic nature may lead the particle to search in regions of the search space

Table 3.1: General PSO Algorithm

A Generic Pseudo Code of Particle Swarm Optimization
Initialize swarm
Evaluate swarm
Identify leader (best particle in the swarm)
While $\mathcal{G} \neq$ TRUE
Fly: update velocity and position
Evaluate swarm
Identify leaders (best own experience and best in the swarm)
End

already visited. [Suganthan 1999] suggested a linear decrease in C_1 and C_2 but reported no improvement in performance by using this scheme. [Ratnaweera et al. 2002] built further on the suggestion of [Suganthan 1999] and proposed C_1 to be linearly decreasing over time and C_2 linearly increasing. This scheme promotes exploration at early stages of the search and convergence towards the global best near the end of the search. In [Ho et al. 2005] authors argue that r_1 and r_2 are not completely independent and propose the condition $r_2 = 1 - r_1$. [den Berg 2002] showed that the basic PSO is neither a local nor global optimizer because once the search reaches the state where $\vec{x}_i = \vec{x}_{BO_i} = \vec{x}_{BG}, \forall i = 1, 2, ..., n_{swarm}$, no further progress can be made. The problem is that if this state is reached before the optimum is found the swarm will be stuck. The author in [den Berg 2002] suggested two options to extend PSO to a global search algorithm. The first option is to generate new random solutions, for example with a mutation operator, or force a random search around $\vec{x}_{BG}$ with hill-climbing or a similar approach [Engelbrecht 2005]. The second option is to *multi-start* PSO, i.e. once the swarm has converged the particles are re-initialized randomly.

The operator mutation, sometimes called turbulence or *craziness*, is intended to promote diversity in the swarm and avoid –or rescue from– premature convergence to potential local minima. This operator has significant impact in the performance of the algorithm and even though it is normally applied in the decision variable space some authors such as [Ho et al. 2005] have applied it to the velocity vector $\vec{v}$ just before updating the position of a particle.

PSO was originally developed for continuous-valued spaces. The first discrete PSO to operate on binary search spaces was developed in [Kennedy and Eberhart 1997]. In practice, the types of problems that can be solved with PSO are large-scale, discrete, combinatorial, and non-convex problems. PSO avoids local optima by using specific operators and does not require restrictive conditions in the function such as continuity or differentiability to the second order.

PSO for Multi-Objective Optimization

One of the most important issues in **multi-objective optimization** (MOO) in general and also in MOOPSO is that of determining when one solution is better than another solution with respect to all objectives. In the single-objective case $\vec{x}_{BO}$ is replaced by $\vec{x}$ if $f(\vec{x}) < f(\vec{x}_{BO})$, where f is the only objective function in the optimization (minimization) problem. But in the multi-objective case this replacement is not that intuitive and notions of preference, such as those of Pareto-dominance introduced in Section 2.1, are needed.

As stated in Section 2.1, MOO algorithms should maintain a record of the already obtained solutions and also promote diversity of solutions. The research work on this thesis makes use of the *adaptive archive grid* approach introduced in [Coello and Salazar-Lechuga 2002] and [Coello et al. 2002], and later improved in [Coello et al.

2004], to manage the truncated archive[7] of non-dominated solutions. The adaptive archive grid approach is based on the *Pareto archive evolutionary strategy* (PAES) proposed in [Knowles and Corne 2000], in which the objective function space is divided into a certain number of hypercubes. The edge of the hypercubes is calculated by $l = \alpha \frac{f_{k_{max}} - f_{k_{min}}}{n_{swarm}}$, $\forall k = 1, 2, ..., n_{obj}$, where $\alpha \in [0, 1]$ is the selection pressure, $f_{k_{max}}$ and $f_{k_{min}}$ are the maximum and minimum values of f_k in the archive respectively, and n_{swarm} is the size of the swarm. A new particle is added into the archive if it is a non-dominated solution. Each non-dominated solution is assigned to its corresponding hypercube according to its $\vec{f}$ value, where the value of each of the objectives is a coordinate in the hyperdimensional space.

Among the most important features of the multi-objective PSO approaches are the mechanisms used to select the leader $\vec{x}_{BG}$ (Eq. 3.2) from the archive. The adaptive archive grid algorithm in [Coello et al. 2004] assigns a fitness value to each hypercube that contains more than one particle. The fitness value of each hypercube is calculated as $f_h = \frac{k}{n_d}$, where k is any number $k > 1$, and n_d is the number of particles contained in that hypercube. This function aims to decrease the fitness of the hypercubes with higher density. $\vec{x}_{BG}$ is then selected from the archive using the roulette wheel selection on the hypercubes fitness values. If the selected hypercube has $n_d > 1$ then $\vec{x}_{BG}$ is selected randomly among the particles contained in that hypercube. If the maximum capacity of the archive has been reached and a new particle has to be added, then a

[7]By truncated we mean that the archive has a maximum capacity.

particle from a highly populated area of the Pareto front is removed.

The Pareto dominance criteria is used to update the value of $\vec{x}_{BO}$, in other words, if the current position is dominated by $\vec{x}_{BO}$, then $\vec{x}_{BO}$ remains intact. If the current position dominates $\vec{x}_{BO}$, then $\vec{x}_{BO}$ is replaced by the current position of the particle. If neither position dominates one of them is selected randomly. If a particle that is a member of the archive becomes dominated it is removed.

3.3 Distributed Search Systems

Multi-agent systems are an approach to model and implement distributed search concepts. The following terminology is taken from [Denzinger 2000]. A **multi-agent search system** $\mathcal{MASS}$ consists of a start agent $\mathcal{A}g_S$, an end agent $\mathcal{A}g_E$, a set of search agents $\{\mathcal{A}g_1, \mathcal{A}g_2, ..., \mathcal{A}g_{n_{\mathcal{MASS}}}\}$, and a communication structure $\mathcal{K}om$. The objective of a $\mathcal{MASS}$ is to solve a specific search problem. The search agents reside in a set of processing units which could be processors, cores, computers, etc.. The role of the start agent $\mathcal{A}g_S$ is to take an instance Ins of the search problem and create instances Ins_1, Ins_2, ..., $Ins_{n_{\mathcal{MASS}}}$, one for each search agent in the $\mathcal{MASS}$. The instances Ins_i, $(\forall i = 1, 2, ..., n_{\mathcal{MASS}})$ can be exact copies of the original Ins, portions of it, or rather different problems related to Ins. Each search agent works on the search instance it received from $\mathcal{A}g_S$ and communicates, with certain frequency, with the rest of the search agents using the communication structure $\mathcal{K}om$. Once the search of the search agents has reached a point that allows the generation of a solution the end agent

$\mathcal{A}g_E$ creates and reports the solution to Ins found by the $\mathcal{MASS}$.

3.3.1 A Search Agent

A **search agent** $\mathcal{A}g_i$ is characterized by the triple $(\mathcal{P}_i, \mathcal{K}om, mes_i)$, where $\mathcal{P}_i$ is a search process, and mes_i is the communication function of $\mathcal{A}g_i$. As stated in Section 3.1, $\mathcal{P}_i$ itself is characterized by the triple $(\mathcal{A}_i, \mathcal{E}nv_i, \mathcal{K}_i)$, where $\mathcal{A}_i$ is the search model used by the process, $\mathcal{E}nv_i$ is the environment within the $\mathcal{MASS}$ that the search process perceives, and $\mathcal{K}_i$ is the search control that $\mathcal{P}_i$ uses for $\mathcal{A}_i$. The environment is a subset of the data areas in $\mathcal{K}om$. The search control selects the next step of the search (s^{new}) based on the current state (s) and the current values of the data areas in the environment (e). Since e is considered by the search control, the search of a search agent can be influenced by changing its environment e. Other search agents can change e by using their communication functions. A communication function mes_i takes the current values of the data areas in $\mathcal{K}om$ and the current search state of $\mathcal{A}g_i$ and creates new values for the data areas in $\mathcal{K}om$. It usually creates only new values for some of the data areas, not all of them.

Two important parts of the definition of a $\mathcal{MASS}$ are the structure of $\mathcal{K}om$ and the effect that the functions mes_i have over $\mathcal{K}om$. The communication structure $\mathcal{K}om$ can be used to model approaches such as blackboard-like and message-passing, but it should be noted that implementing a particular $\mathcal{K}om$ on different hardware structures can lead to considerable differences in performance. This can also be true for the method in

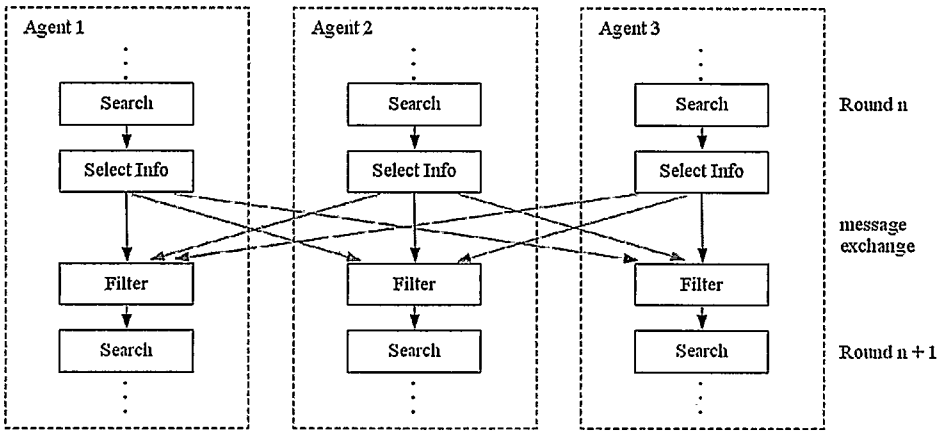
which the search agents are mapped to the set of processing units. If performance is an issue for the search problem at hand, then the best option is to assign each active search agent to its own processing unit. As a consequence the number of active search agents should not exceed the number of processing units available in the system.

3.3.2 Our Distribution Concept

The distributed search approach used in this work is based on a homogeneous version of TECHS (TEams for Cooperative Heterogeneous Search) introduced in [Denzinger and Fuchs 1999] and [Denzinger and Offermann 1999]. TECHS is an instantiation of the *Improving on the Competition Approach* paradigm from [Denzinger 2000], in which every search agent is provided with a complete instance of the search problem to be solved. In the pure *Competition Approach* the search agents simply perform the search in parallel until one of them finds a solution. The performance of the *Competition Approach* can be improved by allowing the team of search agents to communicate during the course of the search and share important information to guide the search.

In TECHS, the work of the search agents is done in so-called rounds, and each round consists of three phases: search, information selection, and filtering. During the search phase each search agent tries to solve the received search instance using its search process. During the information selection phase each search agent evaluates its results and decides whether these results might be of interest to some or all other search agents in the $\mathcal{MASS}$. The selected information is sent out to these other search

Figure 3.1: A round in TECHS for 3 search agents.



agents, which in turn evaluate the usefulness of the received information and, if useful (filtering phase), they integrate the information into their search for the next round. Figure 3.1 illustrates one round of TECHS for three search agents.

There are two slight differences between the distributed approach proposed in this research work and TECHS. The first difference resides in the message exchange, and the second in the filtering phase. As in TECHS, all search agents perform the search, then each of them selects the important information to be shared with the rest of the search agents in the $\mathcal{MASS}$. But instead of all search agents sending the selected information to everyone, only $n_{\mathcal{MASS}} - 1$ search agents send their selected information to a master agent $\mathcal{A}g_M$. $\mathcal{A}g_M$ collects the information, including its own, performs the filtering phase, and sends back the filtered information to the rest of the search agents in the $\mathcal{MASS}$. The search agents in the $\mathcal{MASS}$ integrate the received information and continue with the next round of the search until the termination condition is reached.

The communication structure $\mathcal{K}om$ has designated data areas for each search agent. There are two main types of information search agents can communicate: partial solutions[8] and control information. Each of these types of information can, in turn, be classified as positive information or negative information. Partial solutions are directly incorporated into the search state of a search agent, while control information affects the search control of a search agent [Denzinger and Offermann 1999].

The *mes*-function of a search agent is used to select the information to be shared with other search agents. The *mes*-function is allowed to modify the data areas in $\mathcal{K}om$ associated with the other search agents in the $\mathcal{MASS}$. The *mes*-functions are usually implemented by sending messages to other search agents. The integration of new information into the data areas for an agent $\mathcal{A}g_i$ that forms the agent's environment $\mathcal{E}nv_i$ is done according to the type of information. As already stated, solutions are appropriately added into the search state if they pass the filter criterion, while control information influences the decisions of $\mathcal{K}_i$ of $\mathcal{A}g_i$.

The whole search ends if either: the solution is found –if it is possible for the search agents to detect this– or when a certain time limit is reached. $\mathcal{A}g_E$ collects the results from all search agents and then either presents the solution or the best result found so far in the given time frame.

[8]Note that the definition of partial solutions includes full solutions.

4 Optimization of Natural Gas Pipeline Operations using Multiple Cooperating Set-based Search Agents

This chapter introduces the proposed multi-agent cooperative search approach applied to the solution of the complex problem of optimization of natural gas pipeline operations in which the objective is to supply the desired demand at delivery locations with adequate pressure satisfying the governing laws of conservation of mass and energy.

To demonstrate the performance of the proposed approach we tested it using real -and as consequence complex- problem instances provided by TransCanada PipeLines Ltd. The experimental evaluation shows that our approach meets the realistic time requirements imposed by the transportation industry and demonstrates that it reliably outperforms the interactive method currently used in commercial tools. The proposed search system is not only fast but also capable of finding better solutions than state-of-the-art methods currently used by the natural gas transportation industry. Results of the experiments reported in Chapter 7 show that, in terms of fuel consumption, the cost of our solutions for large sub-networks are at least 12% better than the compared

approaches. For significant volume of transported gas this percentage translates into a considerable monetary amount. Now, it is important to mention that reduction of the energy used in pipeline operations not only has a tremendous economical impact but also an environmental one. More efficient operation of compressor stations results in less greenhouse gas emissions being dissipated to the atmosphere.

This chapter is organized as follows. Section 4.1 introduces particle swarm optimization (PSO) as an instantiation of set-based search, Section 4.2 presents a distribution concept for set-based PSO, and Section 4.3 details the application of the distributed set-based PSO approach to the optimization of NGPO.

4.1 PSO as Set-based Search System

This section presents particle swarm optimization as an instantiation of set-based search by merging the terminology of set-based search systems introduced in Section 3.1 with the principles of PSO introduced in Section 3.2.

4.1.1 Search Model

As stated earlier, a set-based **search model** $\mathcal{A}$ is defined by a set of facts $\mathcal{F}$ and a set of extension rules $\mathcal{E}xt$. In the instantiation of set-based search to describe PSO, a **particle** p is described by:

$$p = (\vec{x}, \vec{v}, \vec{x}_{BO}) \tag{4.1}$$

where $\vec{x}$ is the vector of decision variables of the optimization problem as stated in Eq. 2.1, also called the position of the particle p, $\vec{v}$ is the velocity vector of particle p, and $\vec{x}_{BO}$ is the particle's best position found so far. The **set of facts** $\mathcal{F}$ is the set of all possible particles defined as $\mathcal{F} = \{(\vec{x}, \vec{v}, \vec{x}_{BO}) | \vec{x}, \vec{x}_{BO} \in \mathcal{SS}, \vec{v} \in \mathcal{VS}\}$ where $\mathcal{SS}$ is a predefined solution space limiting the values that decision variables can take, and $\mathcal{VS}$ is a predefined velocity space set to control the global exploration of particles, i.e. to avoid divergence.

As stated in Section 3.2, the idea behind PSO is to update the position of each of the particles of the swarm in every step of the search, i.e. by letting the swarm fly. The new position of each particle is influenced by its own experience and by the experience of the rest of the particles in the swarm. For that reason, in order to update the position of a particle during its flight it is necessary to identify first the best position of each particle i ($\vec{x}_{BO_i}, \forall i = 1, 2, ..., n_{swarm}$), and the best position found so far by the whole swarm ($\vec{x}_{BG}$). With these two values in hand the new velocity ($\vec{v}^{new}$) and new position ($\vec{x}^{new}$) of each particle can be updated following Equations 3.2 and 3.1 respectively.

Now, using the set-based search terminology, the flight of the swarm is governed by a set of **extension rules** $\mathcal{E}xt$. The set $\mathcal{E}xt$ works on a complete search state and dictates the valid transitions to move the particles of the swarm from one state of the search (s) to a new one (s^{new}). The set of valid extension rules is defined as follows:

$$\mathcal{E}xt : \{\{p_1, p_2, ..., p_{n_{swarm}}\} \rightarrow \{p_1^{new}, p_2^{new}, ..., p_{n_{swarm}}^{new}\} \mid$$

$$p_i = (\vec{x}_i, \vec{v}_i, \vec{x}_{BO_i}),\ p_i^{new} = (\vec{x}_i^{new}, \vec{v}_i^{new}, \vec{x}_{BO_i}^{new}),\ \forall i = 1, 2, ..., n_{swarm}$$

$$\vec{x}_{BO_i}^{new} = \begin{cases} \vec{x}_i & \text{iff } \vec{x}_i \lhd \vec{x}_{BO_i} \\ \vec{x}_{BO_i} & \text{else} \end{cases}$$

$$\vec{v}_i^{new} = \mathcal{W}\vec{v}_i + C_1 r_1(\vec{x}_{BO_i}^{new} - \vec{x}_i) + C_2 r_2(\vec{x}_{BG}^{new} - \vec{x}_i) \text{ for some } r_1, r_2 \in [0, 1]$$

$$\vec{x}_i^{new} = \vec{x}_i + \vec{v}_i^{new}\ \}$$

$$\cup$$

$$\{\{p_1, p_2, ..., p_{n_{swarm}}\} \rightarrow \{p_1^{new}, p_2^{new}, ..., \tilde{p}_j^{new}, ..., p_{n_{swarm}}^{new}\} \mid$$

$$p_i = (\vec{x}_i, \vec{v}_i, \vec{x}_{BO_i}), \forall i = 1, 2, ..., n_{swarm},$$

$$\tilde{p}_j^{new} = (\vec{x}_j^{new}, \vec{v}_j^{new}, \vec{x}_{BO_j}^{new}), \vec{x}_j^{new} = mut(\vec{x}_j), \vec{v}_j^{new} = \vec{v}_j,\ 1 \leq j \leq n_{swarm}$$

$$\vec{x}_{BO_j}^{new} = \begin{cases} \vec{x}_j^{new} & \text{iff } \vec{x}_j^{new} \lhd \vec{x}_{BO_j} \\ \vec{x}_{BO_j} & \text{else} \end{cases}$$

$$p_i^{new} = p_i,\ \forall i \neq j\}$$

The symbol $\lhd$ is a criterion[9] used to compare two vectors (in this case vectors that represent positions) and indicate which one is better, i.e. $\vec{x}_1$ is better than $\vec{x}_2$ if $\vec{x}_1 \lhd \vec{x}_2$. This operator is defined according to the problem to be solved. $mut(\vec{x})$ indicates a mutation to the vector $\vec{x}$ which is also defined according to the problem to be solved.

[9]Ordering, ranking, preference, etc.

The **mutation operator** used in this work is the so-called non-uniform mutation [Michalewicz 1996], in which the variability range $\mathcal{R}_i$ $(i = 1, 2, ..., n_x)$, of each decision variable decreases over time. The mutation operator modifies the value of the decision variables of a particle with certain probability $P_{mut} \in [0, 1]$. P_{mut} is the probability of mutation, normally a user-defined value that is kept constant. More specifically, the variability range $\mathcal{R}_i$ of decision variable i is narrowed down during the flight of the swarm according to the non-linear function $\mathcal{R}_i = (1 - g/\mathcal{G})^{5/P_{mut}}$ proposed in [Coello et al. 2004], such that $\mathcal{R}_i \rightarrow 0$ as $g \rightarrow \mathcal{G}$, where g is the current transition, and $\mathcal{G}$ is the termination condition for the search expressed as maximum number of transitions.

The values of $\vec{x}_{BG}^{new}$, $\mathcal{W}$, C_1, and C_2 are explained in Section 4.1.3. By updating the values $\vec{x}_i^{new}$, $\vec{v}_i^{new}$, and $\vec{x}_{BO_i}^{new}$ for each particle the whole swarm flies over the search space towards the optimal solution $\vec{x}^*$.

4.1.2 Search Instance

A **search instance** Ins is defined by an initial search state s^{init} and a termination condition $\mathcal{G}$ as:

$$Ins = (s^{init}, \mathcal{G}) = (\{p_1^{init}, p_2^{init}, ..., p_{n_{swarm}}^{init}\}, \mathcal{G}) \tag{4.2}$$

s^{init} can be generated randomly or by using knowledge related to the problem to solve and the particular instance. Note that at the beginning of the search each particle p_i is described by $p_i^{init} = (\vec{x}_i^{init}, \vec{v}_i^{init}, \vec{x}_i^{init})$ $(\forall i = 1, 2, ..., n_{swarm})$ because at the beginning of the search $\vec{x}_{BO_i}^{init} = \vec{x}_i^{init}$. The criterion to create s^{init} will definitely have

an impact on the quality of the solution of the search, for example an inappropriate criterion could direct the swarm towards a local minimum, an early convergence. $\mathcal{G}$ can be set to a maximum number of particle evaluations or a processing time limit.

4.1.3 Search Process

As introduced in Section 3.1, a **search process** is defined as $\mathcal{P} = (\mathcal{A}, \mathcal{K}, \mathcal{E}nv)$. In the instantiation of set-based search to PSO the search control $\mathcal{K}$ makes use of explicit information to guide the flight of the swarm. $\mathcal{K}$ stores the best solution found so far by the whole swarm, $\vec{x}_{BG}$. This value is calculated as follows:

$$\vec{x}_{BG}^{new} = \begin{cases} \vec{x}_{BO_i}^{new} & \text{iff } \vec{x}_{BO_i}^{new} \lhd \vec{x}_{BG}, \text{ and } \neg\exists j\ \vec{x}_{BO_j}^{new} \lhd \vec{x}_{BO_i}^{new}, (i \neq j), i, j = 1, 2, ..., n_{swarm} \\ \vec{x}_{BG} & \text{else} \end{cases}$$

where $\lhd$ is defined as in Section 4.1.1.

The search control $\mathcal{K}$ also selects the inertia weight $\mathcal{W}$, and the social C_1 and cognitive C_2 learning factors. As mentioned earlier, there already exist in literature several methods to select these parameters. In this thesis we use the criterion introduced in [Toscano-Pulido and Coello 2004]. This criterion is an approach useful to deal with the difficulties of fine tuning of these parameters for specific problems or problem instances. The parameters are given by $\mathcal{W} = \rho(0.1, 0.5)$ and $C_i = \rho(1.5, 2)$, $i = 1, 2$, where $\rho(l, u)$ is a random number generator function that returns a value within a predefined interval $[l, u]$. $\mathcal{K}$ also selects r_1 and r_2 as follows: $r_i = \rho(0, 1)$, $i = 1, 2$.

The search control $\mathcal{K}$ determines whether the mutation operator is applied to a given particle or not according to: IF $\rho(0,1) < P_{mut}$ THEN the particle is mutated.

In summary, this section presented PSO as an instantiation of set-based search, including the definition of a particle, the initialization of the swarm, the governing rules of the flight, and the termination condition. These concepts are the fundamentals for set-based PSO as search algorithm.

4.2 Our Distributed Set-based PSO Search System

Section 4.1 introduced the sequential (single-agent) definition of set-based PSO. This section introduces the distributed concept of set-based PSO as a collaborative multi-agent search system using the distributed search system concepts introduced in Section 3.3.

Let us assume that our multi-agent search system $\mathcal{MASS}$ consists of $n_{\mathcal{MASS}}$ search agents each of which is assigned to its own processing unit[10]. One of the search agents has the role of master agent $\mathcal{A}g_M$ which in addition to solving its own search instance it also performs the role of start agent ($\mathcal{A}g_S$) and end agent ($\mathcal{A}g_E$) as defined in Section 3.3. The master agent $\mathcal{A}g_M$ assigns a search instance Ins_i to each $\mathcal{A}g_i$ in the $\mathcal{MASS}$, including itself.

[10] As stated earlier, the number of processing units (n_{pu}) available in the system should be greater or equal than the number of search agents in the $\mathcal{MASS}$, i.e. $n_{pu} \geq n_{\mathcal{MASS}}$, to improve the performance of the search.

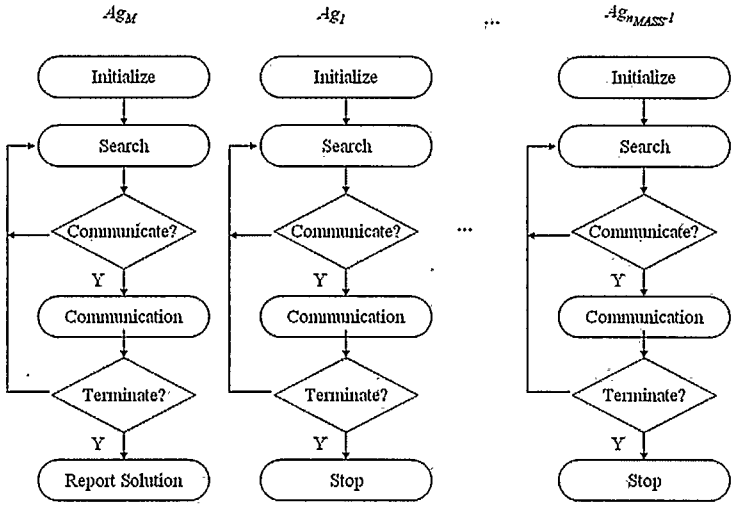
As in the case of most set-based search methods, in our distributed set-based PSO the initial state of each search agent $\mathcal{A}g$ consists of a set of randomly created particles, i.e. each $\mathcal{A}g$ initializes its own swarm resulting in $n_{\mathcal{MASS}}$-different initial swarms or states.

Once all search agents are initialized they proceed to perform the search. Every time the communication condition $\mathcal{G}_{comm}$ becomes TRUE the communication is started and each search agent $\mathcal{A}g_i$ ($\mathcal{A}g_i \neq \mathcal{A}g_M$) sends its selected information to the master agent $\mathcal{A}g_M$. The communication condition $\mathcal{G}_{comm}$ is set TRUE every n_{comm}-number of search steps (see Figure 4.1). Although this condition imposes synchronization, a potential bottleneck in the system, and may let one or more search agents be idle for a little time, the proposed communication condition proved to be appropriate for the search instances tested leading to insignificant idle time[11].

The distributed set-based PSO makes use of positive information in the following manner. $\mathcal{K}om$ contains two types of data areas called **positive control information** (k_i^+) and **positive solution information** (sol_i^+) for each search agent $\mathcal{A}g_i$ in the $\mathcal{MASS}$. In k_i^+, the other search agents' mes_j-functions ($i \neq j$, $i, j = 1, 2, ..., n_{\mathcal{MASS}}$) put their best solution found so far: $\vec{x}_{BG_j}$. In sol_i^+, the other search agents' mes_j-functions put the best k_1 solutions they found during the last round. At first glance the information in k_i^+ seems to be redundant but is not because of two reasons. Both

[11]This communication scheme works well to solve problems in which the computational effort for communication is negligible compared to the run time for the evaluation of solutions to avoid overloading $\mathcal{A}g_M$ and slow down the search.

Figure 4.1: Communication between search agents in the $\mathcal{MASS}$.



reasons are related to the use that $\mathcal{A}g_i$ makes of this information in its own environment.

First, the information in k_i^+ is filtered by selecting the best solution overall. If the best solution overall is better than the search agent's $\vec{x}_{BG}$ then the search agent updates its $\vec{x}_{BG}$ with the new value just received. This indicates that information in k_i^+ affects the search control $\mathcal{K}_i$ of the search agent $\mathcal{A}g_i$, which in turn affects the adjustment of the velocity vector $\vec{v}$. Now, the information in sol_i^+ is also filtered by selecting the best k_2 solutions in it. The k_2-selected solutions are then used to replace the k_2 worst current solutions in each $\mathcal{A}g_i$'s current state essentially moving the position of these k_2 particles to different, hopefully better, areas of the search space. It should be noted that the velocity vectors of these k_2 particles are not affected by the substitution and

that the best ancestor of each particle is only updated if the new solution is better. By not influencing the velocity vectors, different search agents might have particles located at the same coordinates in the search space but will produce different successors if the velocities and the leaders are different. In conclusion, the use that search agents make of the k_i^+ and sol_i^+ data areas is definitely different.

The second reason makes the need for the two data areas even more evident. The values of sol_i^+ $(i = 1, 2, ..., n_{MASS})$ are not to be updated after every single round. They are updated only after every $n_{sol_i^+}$ rounds. If the search states of all search agents were to be updated after every single round this would quickly direct all search agents to explore the same area of the search space thus resulting in redundant and perhaps useless work. While the use of k_i^+ only establishes a common direction for the particles, using sol_i^+ too often would move k_2 particles to, or near to, this best solution without exploring solutions situated between the current positions and this best area. It should be noted that sol_i^+ has the capability to rescue search agents currently exploring bad areas of the search space by moving the particles elsewhere. Balancing the frequency of the use of sol_i^+ requires some calibration of the system and this is how $n_{sol_i^+}$ was determined in the experiments of this thesis.

4.3 Distributed Set-based PSO for Optimization of Pipeline Operations

This section describes the application of our distributed set-based PSO search system to the optimization of operations of natural gas pipeline networks. The end result of this section is a system called IOPO, which stands for Intelligent Optimization of Pipeline Operations.

4.3.1 Optimization of NGPO

As indicated in Chapter 2, optimization of NGPO can be achieved by optimizing objectives such as fuel consumption, throughput and linepack. Without loss of generality these objectives are represented here as f_1, f_2, and f_3 respectively. Following Equation 2.1, the NGPO optimization problem can be stated as finding the vector $\vec{x}$ which:

$$Minimize\ \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})] \tag{4.3}$$

subject to

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, ..., n_{ineq} \tag{4.4}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., n_{eq} \tag{4.5}$$

where $\vec{x}$ is a vector of **decision variables** of dimension n_x whose components are the parameters needed to operate a specific pipeline network. As mentioned in Chapter 2, due to the nature of the NGPO problem the vector $\vec{x}$ includes a mix of continuous and discrete decision variables, i.e. $\vec{x} = [x_1^d, x_2^d, ..., x_{n_d}^d, x_1^c, x_2^c, ..., x_{n_c}^c]$, where n_d and n_c

are the number of discrete (x_i^d) and continuous (x_i^c) decision variables respectively, and $n_x = n_d + n_c$. Discrete decision variables in $\vec{x}$ are the status of each compressor station (ONLINE/OFFLINE) and the state of each block valve (FULLYOPENED/FULLYCLOSED) present in the pipeline network. The continuous decision variables in $\vec{x}$ are control pressures at each compressor station (either P_s or P_d), control pressure at each control valve (either P_s or P_d), and the control pressure at each meter station considered in the pipeline system under consideration. An additional continuous decision variable is the proration ratio used to modulate the volume of gas received/delivered in the pipeline system.

Discrete decision variables are treated as continuous variables during the flight of the swarm, i.e. $x_i^d \in [0, 1]$. Then, just before $\vec{x}$ is evaluated the values of x_i^d are rounded to discrete values, $x_i^d \in \{0, 1\}$, to represent the ONLINE, OFFLINE, FULLYOPENED, or FULLYCLOSED value. The values that the continuous decision variables are allowed to take are restricted by the physical limits of the pipeline network, e.g. maximum/minimum allowed operating pressure, minimum contractual delivery pressure, etc.. For example, in case a decision variable represents the pressure at a control node the decision variable will be allowed to take only values within a specific range of pressures for design and safety reasons. Specifically, $x_i^c \in [P_{min}, P_{max}]$, where P_{min} and P_{max} are the minimum and maximum allowed pressure values for this element or device within the pipeline system.

For the NGPO problems studied in this thesis, the following elements of a pipeline network were considered:

Compressor stations. For each CS two decision variables are needed: one continuous decision variable to represent the suction or discharge pressure, and one discrete decision variable to represent the status of the CS.

Block valves. For each BV present in the pipeline network one discrete decision variable is necessary to indicate its status.

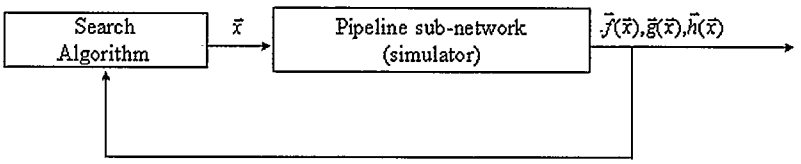
Control valves. For each CV present in the pipeline network one continuous decision variable is necessary to indicate its control pressure.

Meter stations. For each meter station present in the pipeline network one continuous decision variable is necessary to indicate the pressure at this control node.

Proration ratio. One continuous decision variable is used to represent the proration factor at a specific meter station.

$\vec{f}(\vec{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^3$ is the vector of **objective functions**, $f_1(\vec{x})$ is the amount of fuel consumed by all ONLINE compressor stations in the pipeline network, $f_2(\vec{x})$ is the throughput at a specific node or nodes of the network, and $f_3(\vec{x})$ is the total linepack in the pipeline system. The functions $\vec{g}(\vec{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_{ineq}}$ and $\vec{h}(\vec{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_{eq}}$ are constraint functions of the NGPO problem at hand. The value of $\vec{f}(\vec{x})$ is calculated using a hydraulic network solver, which in the following will be called **hydraulic simulator**. The hydraulic simulator provides few results which can be used to assess

Figure 4.2: Communication between the search algorithm and the hydraulic simulator



the quality of each candidate solution vector $\vec{x}$. The hydraulic simulator used in the experiments of the NGPO problem is a non-isothermal hydraulic model that simulates the steady state gas flow within the pipeline system considering temperature profiles and composition of the transported gas. The hydraulic simulator basically evaluates how well the pipeline system would operate if the values of the parameters contained in $\vec{x}$ were to be implemented to operate the modeled pipeline network. The hydraulic simulator returns measures of performance such as the total fuel consumption by compressor stations, the throughput at strategic nodes, and the total linepack in the system (see Figure 4.2).

The hydraulic simulator used in the experiments of this research work is proprietary to TransCanada PipeLines Ltd. and therefore had to be treated like a blackbox. It can only be disclosed that, at very high level, the search process calls the hydraulic simulator every time it needs to evaluate a solution vector $\vec{x}$. The solution vector is written into a file which is part of the input information required by the hydraulic simulator. The hydraulic simulator does its work and then writes the results of the simulation into an output file. This file is read by the search process after the hydraulic

simulation ends.

The hydraulic simulator reports all the **constraint violations** caused by $\vec{x}$ ($\vec{g}(\vec{x})$ and $\vec{h}(\vec{x})$) such as pressure at control nodes with values under P_{min} or over P_{max}, any reverse flow situation, power violations, etc. To keep the reader of this thesis in perspective, the number of n_{ineq} and n_{eq} is in the order of hundreds to thousands for the problem instances studied in this research work.

The hydraulic simulator also returns what we call the **hydraulic correctness** of a solution vector, an integer value $\mathcal{HC}(\vec{x}) \in \{0, 1\}$. If $\mathcal{HC}(\vec{x}) = 1$, $\vec{x}$ is assumed to be hydraulically correct. Note that $\vec{x}$ may be an infeasible solution vector, i.e. it may cause constraint violations within the pipeline system such as pressure violations or reverse flows, but the hydraulic simulator is able to converge to a solution. If $\mathcal{HC}(\vec{x}) = 0$, then $\vec{x}$ is considered hydraulically incorrect, i.e. the vector $\vec{x}$ contains operational settings that are not only infeasible but physically impossible to be implemented in the pipeline network modeled.

4.3.2 Set-based PSO for Optimization of NGPO

A **solution** for an instance of the NGPO problem consists of the parts of the vector $\vec{x}$ that represent the parameters needed to operate a specific pipeline network as defined in Section 4.3.1. An **instance** of the NGPO problem consists of the network topology that is needed for the pipeline simulator, the technical description of all compressor units at the CSs, the BVs, the CVs, the flow requirements at specific points in the

pipeline network, and the set of boundary conditions for the network that describe receipts and deliveries, i.e. location and volume of gas that goes into or out of the network.

A **particle** p for an instance of the NGPO problem is described by $p = (\vec{x}, \vec{v}, \vec{x}_{BO})$ where $\vec{x}$ is the vector of decision variables as described in Section 4.3.1, $\vec{v}$ is the vector of velocities of the particle, and $\vec{x}_{BO}$ is the particle's best position found so far. $\mathcal{SS}$ is by default restricted by the physical limits of the pipeline components. $\mathcal{VS}$ is limited to prevent the velocity from growing out of bounds. Traditionally, in set-based search systems the initial values of $\vec{x}$ are selected randomly within $\mathcal{SS}$. We found that $\mathcal{SS}$ can in fact be reduced to a significantly smaller set $\mathcal{SS}^{init}$ ($\mathcal{SS}^{init} \subset \mathcal{SS}$) in such a way that the time needed to perform the search and find an acceptable solution is remarkably less, thus improving the performance of the search algorithm.

4.3.2.1 Using NGPO Knowledge to Reduce the Search Space

As explained in Chapter 3, general knowledge about the application problem can be considered in the design of the search model $\mathcal{A}$ and/or search control $\mathcal{K}$ to improve the performance of the search. It should be noted that selection of the knowledge is critical: appropriate knowledge can improve the search but, by the same token, badly chosen knowledge can weaken the performance of the search. Useful knowledge about the NGPO problem was identified and used to improve the efficiency of the proposed set-based PSO search system. It is important to emphasize that the NGPO knowledge

used to improve the proposed search approach is generic knowledge and is applicable to all NGPO problem instances tested in this thesis. It is knowledge that is available or, in the worst case, not too complicated to obtain. This knowledge revealed certain properties of the NGPO problem.

We used part of the acquired knowledge about the NGPO problem to improve the design of the search model by restricting the initial solution space from $\mathcal{SS}$ to $\mathcal{SS}^{init}$ to speed up the search. We observed that, in contrast to other real-world optimization problems, solutions to the NGPO problem (as stated in this research work) that are close in the objective space are also close in the decision variable space. Based on this information, it is assumed that once an acceptable solution is found there is a high probability that the optimal solution is not too far from there.

This knowledge is used in the following way. The initial population is generated with the position of the particles initialized randomly within $\mathcal{SS}$, i.e. $\vec{x}_i^{init} \in \mathcal{SS}, \forall i = 1, 2, ..., n_{swarm}$, with zero velocity. And because $\vec{x}_{BO_i}^{init} = \vec{x}_i^{init}$ then we represent each initial particle as $p_i^{init} = (\vec{x}_i^{init}, \vec{0}_i, \vec{x}_i^{init})$[12]. The search, as described before, is performed until the termination condition $\mathcal{G}$ is set TRUE. $\mathcal{G}$ becomes TRUE when the first **acceptable solution** is found. For the NGPO problem, the first acceptable solution is the first solution vector $\vec{x}$ that the hydraulic simulator reports as being hydraulically correct ($\mathcal{HC} = 1$)[13]. The time needed to find an acceptable solution varies with the complexity and size of the pipeline network, i.e. it is faster to find acceptable solu-

[12]The velocity vector is updated first, then the position vector.

[13]The term hydraulic correctness was introduced in Section 4.3.1.

tions for small pipeline networks than for large and complex ones. The solution to this problem at this particular stage of the search, i.e. the first acceptable solution found, is represented by $\vec{x}_0$.

The position vector $\vec{x}_0$ becomes the *center* of what will be called $\mathcal{SS}^{init}$, delimited by $\vec{x}_0 \pm \alpha\vec{\mathcal{R}}$, where α is a parameter $\alpha \in [0, 0.5]$, $\vec{\mathcal{R}}$ is the vector that contains the variability range of each decision variable, and $\mathcal{SS}^{init} \subset \mathcal{SS}$. Right after the first acceptable solution $\vec{x}_0$ is found the swarm is moved to the neighborhood of $\vec{x}_0$ and the search continues from there. In other words, the positions of all particles are re-initialized randomly with $\vec{x}_i^{init} \in \mathcal{SS}^{init}$, $\vec{v}_i^{init} = \vec{0}$, and

$$\vec{x}_{BO_i}^{init} = \begin{cases} \vec{x}_i^{init} & \text{iff } \vec{x}_i^{init} \lhd \vec{x}_{BO_i} \\ \vec{x}_{BO_i} & \text{else} \end{cases}$$

$\forall i = 1, 2, ..., n_{swarm}$, and $\lhd$ is a comparison operator which will be explained in the next section. Note that none of the components of the newly created vector $\vec{x}_i^{init}$ should take a value outside of the physical requirements imposed by the original $\mathcal{SS}$. A small value of α creates a tight $\mathcal{SS}^{init}$ space around $\vec{x}_0$, and as $\alpha \rightarrow 0.5$, $\mathcal{SS}^{init} \rightarrow \mathcal{SS}$.

The positions of the particles remain limited by the range imposed by $\mathcal{SS}^{init}$ during n_{init}-number of transitions and then the range is reset to the original and wider range defined by $\mathcal{SS}$. Opening the range of the decision variables is important for two reasons. First, to have the opportunity to rescue the swarm in case it becomes trapped in a

bad area of the search space. Second, to have the opportunity to search for better solutions, perhaps the optimum, that might be outside of the space delimited by $\mathcal{SS}^{init}$. The journey of the swarm ends when a predefined number of particle evaluations is reached.

It is important to mention that the statement 'solutions that are close in the objective space are also close in the decision variable space' was observed only on the continuous decision variables ($x_i^c \in [P_{min}, P_{max}]$, $i = 1, 2, ..., n_c$). Discrete decision variables –which could easily account for almost 50% of the total number of decision variables in the NGPO problem– do not necessarily follow this behavior, their value may change abruptly which still gives complexity to the search problem. Otherwise the search would be able to find the optimum solution almost immediately, right after finding $\vec{x}_0$.

4.3.2.2 Criteria to Compare Solutions

In order to update the position and velocity of the particles in the swarm using the set of extension rules introduced in Section 4.1.1 to create the next state of the search, it is necessary to identify the particle's best position found so far ($\vec{x}_{BO}$), and the best position found so far by the whole swarm ($\vec{x}_{BG}$). As a consequence a criterion to compare position vectors is necessary.

If the objective of the optimization problem is to find the vector $\vec{x}^*$ that minimizes the energy consumption for the operation of a given pipeline network, then the obvious

choice is to use the fuel consumption caused by each particle, represented as $f_1(\vec{x})$ in Eq. 4.3, to measure the quality of each solution vector $\vec{x}$. Then, the criterion to compare two solution vectors could be defined as: position 1 is *better* than position 2 (represented as $\vec{x}_1 \triangleleft \vec{x}_2$) if $f_1(\vec{x}_1) < f_1(\vec{x}_2)$. But the obvious selection of using just the fuel consumption to compare solution vectors ignores other information about $\vec{x}$ that is readily available and is important.

As mentioned in Section 4.3.1, each solution vector $\vec{x}$ is evaluated using a hydraulic simulator which determines the effects of using the operational settings encoded in $\vec{x}$ if it were to be implemented in the pipeline network. The most important information reported by the hydraulic simulator, relevant for our research work, is:

- the amount of fuel consumed by the compressor units (f_1), throughput (f_2), and linepack (f_3)
- the physical violations in the pipeline system ($\vec{g}$ and $\vec{h}$)
- the hydraulic correctness of the solution vector ($\mathcal{HC}$)

All this information has influence on the quality of $\vec{x}$ and should be considered when comparing two solution vectors. Current solution approaches may suggest considering all this information simultaneously by creating an evaluation (fitness) function that combines all these metrics into a single numerical value. The fitness of the vector $\vec{x}$ could then be used to compare solution vectors and for control purposes. But finding such a fitness function is not an easy task and often requires additional system pa-

rameters that need to be set for every instance of the search problem to solve. For example, [Botros et al. 2004] and [Botros et al. 2006] combined the value of the objective functions with the physical violations into a cost function and reported difficulty on the dynamic selection of penalty parameters for the violations.

For our approach we have selected a different method, namely the **lexicographical combination of orderings**. In order to compare the position of two particles $\vec{x}_1$ and $\vec{x}_2$, a triple $(\mathcal{HC}(\vec{x}), \Omega(\vec{x}), f_1(\vec{x}))$ is determined for each of them. As introduced in Section 4.3.1, the hydraulic correctness of $\vec{x}$ ($\mathcal{HC}(\vec{x}) \in \{0, 1\}$) is a binary function indicating whether the position vector represents a hydraulically correct solution ($\mathcal{HC}(\vec{x}) = 1$) or not ($\mathcal{HC}(\vec{x}) = 0$). The function $\Omega(\vec{x})$ accounts for all the constraint violations out of the sets $\vec{g}(\vec{x})$ and $\vec{h}(\vec{x})$ as follows $\Omega(\vec{x}) = \sum_{i=1}^{n_{ineq}} |g_i(\vec{x})| + \sum_{i=1}^{n_{eq}} |h_i(\vec{x})|$. And $f_1(\vec{x})$ is the fuel consumed by the compressor units running in all the ONLINE compressor stations. Now, the position of particle 1 is considered better than the position of particle 2 ($\vec{x}_1 \lhd \vec{x}_2$), if either $\mathcal{HC}(\vec{x}_1) > \mathcal{HC}(\vec{x}_2)$, or if $\mathcal{HC}(\vec{x}_1) = \mathcal{HC}(\vec{x}_2)$ and $\Omega(\vec{x}_1) < \Omega(\vec{x}_2)$, or if $\mathcal{HC}(\vec{x}_1) = \mathcal{HC}(\vec{x}_2)$ and $\Omega(\vec{x}_1) = \Omega(\vec{x}_2)$ and $f_1(\vec{x}_1) < f_1(\vec{x}_2)$. In other words, hydraulically correct solutions are always better than hydraulically incorrect ones, solutions with less constraint violations are better than solutions with more constraint violations, and valid solutions are compared based on their energy consumption.

4.3.2.3 Using NGPO Knowledge to Avoid Unnecessary Evaluations

We identified knowledge about the NGPO problem useful to avoid the evaluation of invalid solutions without the need of consulting the time-consuming hydraulic simulation. It is important to mention that the time required by each run of the hydraulic simulator is not constant. After running several experiments it was observed that the time needed by the hydraulic simulator to evaluate a single solution for a given search problem varies tremendously depending on the quality of the solution vector $\vec{x}$. For example, if $\mathcal{HC}(\vec{x}) = 1$ the hydraulic simulator usually needs less time to converge and provide the evaluation results than in the case of $\mathcal{HC}(\vec{x}) = 0$. This reason motivated the characterization of *bad* solution vectors.

It was observed that, for any given instance of the NGPO problem, there is a minimum number of CSs that must be ONLINE to be able to move the given transportation requirements of natural gas. By knowing this information the search agents can easily identify (pre-evaluate) if a particle in the swarm is at a position $\vec{x}$ that does not have the minimum number of CSs running and simply avoid an unnecessary time-consuming evaluation with the hydraulic simulator. It is now known that bad solutions, like the ones that do not satisfy the minimum number of CSs running, consume plenty of time during its evaluation because the hydraulic simulator makes numerous attempts to make the solution converge, and at the end undoubtedly reports that the vector $\vec{x}$ is an invalid solution. A candidate solution that does not satisfy the minimum number of CSs ONLINE will most likely cause a $\mathcal{HC}(\vec{x})= 0$ or a very large $\Omega(\vec{x})$ value.

To make use of this knowledge a pre-evaluation function $\mathcal{M}_{CS}(\vec{x}) \in \{0, 1\}$ is applied to each candidate solution just prior to its evaluation with the hydraulic simulator. If $\mathcal{M}_{CS}(\vec{x}) = 1$ it means that the solution vector $\vec{x}$ satisfies the minimum number of ONLINE CSs, then $\vec{x}$ is evaluated with the hydraulic simulator to generate the triple $(\mathcal{HC}(\vec{x}), \Omega(\vec{x}), f_1(\vec{x}))$. If $\mathcal{M}_{CS}(\vec{x}) = 0$ it means that the solution vector $\vec{x}$ does not satisfy the minimum number of ONLINE CSs, hence this solution vector should not be evaluated with the hydraulic simulator and should be strongly penalized.

4.3.3 A Distributed Set-based PSO for the Optimization of NPGO

This section describes the application of our distributed set-based PSO search approach described in Section 4.2 and illustrated in Fig. 4.1 to the optimization of NGPO. The overall implementation is called IOPO (**I**ntelligent **O**ptimization of **P**ipeline **O**perations).

Like other approaches following the *Improving on the Competition Approach* paradigm as characterized in [Denzinger 2000], IOPO creates one search process for each processing unit (processor or core) available in the system. In other words, there will be as many search agents in the $\mathcal{MASS}$ as processing units are available in the system.

The distributed search starts with the master agent $\mathcal{A}g_M$ assigning a search instance Ins to each of the search agents in the $\mathcal{MASS}$ including itself. For the NGPO problem, each of the $n_{\mathcal{MASS}}$ search agent receives an exact copy of Ins. All search agents have first the task of finding the vector $\vec{x}^*$ of operational settings that consumes the minimum amount of fuel to satisfy the transportation requirements for a specific pipeline network

specified in the search problem.

Initial Population. Each search agent $\mathcal{A}g_i$ ($\forall i = 1, 2, ..., n_{\mathcal{MASS}}$) generates its initial population of particles following the criterion proposed in Section 4.3.2.1, which is based on knowledge about the NGPO problem and designed to speed up the search. Eventually, one of the search agents will find a hydraulic correct solution and will relocate its swarm to that neighborhood. Because all search agents communicate regularly (see Cooperation below), whenever a search agent finds an acceptable solution this solution will be shared to the rest of the $\mathcal{MASS}$ in the next communication step and the particles of all swarms will be relocated. The search agents will no longer search for the first hydraulically correct solution, they now will relocate their population and continue the search from there.

Search. At this point in time all search agents in the $\mathcal{MASS}$ are performing search and evaluating solution vectors using their own instance of the hydraulic simulator.

Cooperation. Every time the communication condition $\mathcal{G}_{comm}$ becomes TRUE, each search agent $\mathcal{A}g_i$ makes use of its mes_i-function to provide its best solution found so far ($\vec{x}_{BG_i}$) into the corresponding data area of $\mathcal{A}g_M$. Among all the $n_{\mathcal{MASS}}$-$\vec{x}_{BG}$s solution vectors there will be one that is the best overall. $\mathcal{A}g_M$ selects the best $\vec{x}_{BG}$ among all and distributes it into the k_i^+ areas. If the best solution overall is *better*[14] than the search agent's $\vec{x}_{BG_i}$ the search agent updates its $\vec{x}_{BG_i}$ with the best overall. There is no exchange of sol_i^+ information during this kind of search.

[14]The criterion used to compare two solution vectors and decide which one is better, if not equal, was introduced in Section 4.3.2.2.

Termination. The $\mathcal{MASS}$ continues the distributed search with the particles of the swarms updating their velocity and position according to the set of extension rules $\mathcal{E}xt$ introduced in Section 4.1.1, establishing communication every n_{comm}-number of search steps until the termination condition $\mathcal{G}$ is set TRUE. $\mathcal{A}g_M$ collects the results of all search agents and presents the best solution to the given search problem.

IOPO balances the danger of having all search processes searching the same area of the search space which typically has as consequence ample redundant computations and the need to get search processes out of not so good (but locally optimal) areas when compared with the areas where the other processes are.

It is important to mention that the evaluation of solution vectors is by far the most computationally expensive part of the optimization of NGPO problem. The evaluation of candidate solutions with the hydraulic simulator takes nearly all the time of each process. In IOPO most processors/cores are running hydraulic simulations in parallel, an effect all known distribution concepts for this type of search want to achieve.

5 Extending IOPO to Multi-Objective Optimization

Most real-life optimization problems have the property of being multi-objective. For single-objective optimization problems the optimum is the best of a set of candidate solutions. But for multi-objective optimization (MOO) the definition of an optimum is not that simple. This is because of the often conflicting objectives, while some solutions may favor one objective at the same time they may deteriorate another objective or objectives. The solution to a MOO problem is a set of non-dominated solutions.

As mentioned in Section 2.1, one of the main goals of every MOO algorithm is to keep record of the non-dominated solutions as the search progresses. Keeping this record can be interpreted as a kind of elitism. At the beginning of the search this record of non-dominated solutions, often called **archive** in MOO literature, is empty. As the search progresses new non-dominated solutions enter into the archive and others are removed. There are two main reasons to remove solutions from the archive: 1) if a solution becomes dominated, and 2) if the archive has limited capacity and a solution has to be removed following a predetermined criterion. The idea is that at the end of

the search the archive will be a subset of the true Pareto front, and as a consequence a subset of the optimal solutions to the search problem. Having a set of solutions being equal in quality (all of them are non-dominated) makes the selection of best particles more complicated than for the single objective case.

This chapter presents the extension of IOPO to the MOO case and it is organized as follows.. First, Section 5.1 extends the set-based PSO search definitions introduced in Section 4.1 now to the MOO case. Then Section 5.2 extends the distributed set-based PSO search definitions introduced in Section 4.2 now to the MOO case. And Section 5.3 presents the application of IOPO to solve the MOO NGPO problem.

5.1 Extending Set-based PSO Search Definitions to MOO

Most of the set-based PSO definitions introduced in Section 4.1 are directly applicable to the MOO case. In this section, each definition is addressed and if there is a difference with the previous definitions it is explained in detail.

5.1.1 Search Model

In set-based PSO, for single and also MOO, a **particle** is defined by the tuple $p = (\vec{x}, \vec{v}, \vec{x}_{BO})$. A set-based **search model** $\mathcal{A}$ is defined by a set of **facts** and a set of **extension rules** ($\mathcal{F} = \{(\vec{x}, \vec{v}, \vec{x}_{BO}) | \vec{x}, \vec{x}_{BO} \in \mathcal{SS}, \vec{v} \in \mathcal{VS}\}$ and $\mathcal{E}xt$ respectively).

The set of extension rules $\mathcal{E}xt$ for the MOO case is given by:

$$\mathcal{E}xt : \{\{p_1, p_2, ..., p_{n_{swarm}}\} \rightarrow \{p_1^{new}, p_2^{new}, ..., p_{n_{swarm}}^{new}\} \mid$$

$$p_i = (\vec{x}_i, \vec{v}_i, \vec{x}_{BO_i}),\ p_i^{new} = (\vec{x}_i^{new}, \vec{v}_i^{new}, \vec{x}_{BO_i}^{new}),\ \forall i = 1, 2, ..., n_{swarm},$$

$$\vec{x}_{BO_i}^{new} = \begin{cases} \vec{x}_i & \text{iff } \vec{x}_i \lhd_{\text{MOO}} \vec{x}_{BO_i} \\ \vec{x}_{BO_i} & \text{else} \end{cases}$$

$$\vec{v}_i^{new} = \mathcal{W}\vec{v}_i + C_1 r_1(\vec{x}_{BO_i}^{new} - \vec{x}_i) + C_2 r_2(\vec{x}_{BG}^{new} - \vec{x}_i) \text{ for some } r_1, r_2 \in [0, 1],$$

$$\vec{x}_i^{new} = \vec{x}_i + \vec{v}_i^{new}\ \}$$

$$\cup$$

$$\{\{p_1, p_2, ..., p_{n_{swarm}}\} \rightarrow \{p_1^{new}, p_2^{new}, ..., \tilde{p}_j^{new}, ..., p_{n_{swarm}}^{new}\} \mid$$

$$p_i = (\vec{x}_i, \vec{v}_i, \vec{x}_{BO_i}), \forall i = 1, 2, ..., n_{swarm},$$

$$\tilde{p}_j^{new} = (\vec{x}_j^{new}, \vec{v}_j^{new}, \vec{x}_{BO_j}^{new}), \vec{x}_j^{new} = mut(\vec{x}_j), \vec{v}_j^{new} = \vec{v}_j,\ 1 \leq j \leq n_{swarm},$$

$$\vec{x}_{BO_j}^{new} = \begin{cases} \vec{x}_j^{new} & \text{iff } \vec{x}_j^{new} \lhd_{\text{MOO}} \vec{x}_{BO_j} \\ \vec{x}_{BO_j} & \text{else} \end{cases}$$

$$p_i^{new} = p_i,\ \forall i \neq j\}$$

which is almost identical to the set $\mathcal{E}xt$ defined for the single-objective case. The only difference is the operator $\lhd_{\text{MOO}}$ used to compare two position vectors and indicate which of the two is better. This operator is defined according to the MOO problem to be solved; our definition is presented in Section 5.3.1. The criterion to select $\mathcal{W}$, C_1, and C_2 is identical to the criterion for the single objective case introduced in Section

4.1.3. The selection of $\vec{x}_{BG}^{new}$ for MOO is explained in Section 5.1.3. The same mutation operator $mut(\vec{x})$ introduced in Section 4.1.1 for the single objective case is used here in the MOO case.

5.1.2 Search Instance

The MOO definition of **search instance** Ins is the same as the one introduced for single objective: $Ins = (s^{init}, \mathcal{G}) = (\{p_1^{init}, p_2^{init}, ..., p_{n_{swarm}}^{init}\}, \mathcal{G})$, where s^{init} is the initial search state and $\mathcal{G}$ is the termination condition of the search. s^{init} can be generated randomly or by using knowledge related to the problem at hand and the particular instance.

5.1.3 Search Process

A **search process** is defined by the tuple $\mathcal{P} = (\mathcal{A}, \mathcal{K}, \mathcal{E}nv)$. As in the single-objective case, in MOO the **search control** $\mathcal{K}$ also makes use of explicit information to guide the flight of the swarm. For single-objective optimization $\mathcal{K}$ stores the best solution found so far by the whole swarm: $\vec{x}_{BG}$. But for the MOO case, instead of storing a single value, $\mathcal{K}$ stores the set of non-dominated solutions found so far by the swarm, $\mathcal{PF}_{temp}$, and selects $\vec{x}_{BG}$ for each particle from that set. To do so, $\mathcal{K}$ performs the *adaptive archive grid* algorithm introduced in Section 3.2. In a few words, in the adaptive archive grid algorithm the objective function space is divided into a certain number of hypercubes. Each element of $\mathcal{PF}_{temp}$ is assigned to an hypercube according

to its $\vec{f}$ value. A fitness value is assigned to each particle in $\mathcal{PF}_{temp}$ which is inversely proportional to the density (or number of particles) contained in the hypercube. The less particles in the hypercube, the higher the fitness value of its elements. $\vec{x}_{BG}$ is selected by $\mathcal{K}$ from the elements in $\mathcal{PF}_{temp}$ using the roulette wheel selection scheme based on the fitness value mentioned before, thus steering exploration towards the hypercubes with less density.

By updating the values $\vec{x}_i^{new}$, $\vec{v}_i^{new}$, and $\vec{x}_{BO_i}^{new}$ for each particle ($\forall i = 1, 2, ..., n_{swarm}$), the swarm will eventually fly towards the Pareto front.

5.2 Extending Distributed Set-based PSO Search to MOO

This section introduces the extension of distributed set-based PSO search to the MOO case. The proposed approach consists of two main phases: PHASE I which aims at finding a preliminary Pareto front with a few transitions, and PHASE II that makes use of the results obtained in PHASE I as starting point to find the Pareto front that optimizes the objectives of the optimization problem at hand. The following two sections present the details of these two phases of IOPO for MOO.

5.2.1 Phase I

During PHASE I the goal of each of the search agents in the $\mathcal{MASS}$ ($\mathcal{A}g_1$, $\mathcal{A}g_2$, ..., $\mathcal{A}g_{n_{\mathcal{MASS}}}$), is to explore a specific area of the search space and obtain, in a limited number of transitions, a set of non-dominated solutions to form a preliminary Pareto

front, denoted as $\mathcal{PF}_{temp_i}$ ($\forall i = 1, 2, ..., n_{\mathcal{MASS}}$), one for each search agent in the $\mathcal{MASS}$.

5.2.1.1 Phase I: Initial Population

The search starts when the master agent $\mathcal{A}g_M$ assigns a particular instance Ins_i to each of the search agents in the $\mathcal{MASS}$. The search instance for each of the search agents $\mathcal{A}g_i$ is defined by $Ins_i = (s_i^{init}, \mathcal{G})$, where $\mathcal{G}$ is a predefined number of transitions, $s_i^{init} = \{p_{1,i}^{init}, p_{2,i}^{init}, ..., p_{n_{swarm},i}^{init}\}$, $p_{j,i}^{init} = (\vec{x}_j^{init}, \vec{v}_j^{init}, \vec{x}_{BO_j}^{init}) = (\vec{x}_j^{init}, \vec{0}, \vec{x}_j^{init})$, and $\vec{x}_j^{init} \in \mathcal{SS}_i^{init}$, $\forall i = 1, 2, ..., n_{\mathcal{MASS}}$, $\forall j = 1, 2, ..., n_{swarm}$. As mentioned before, the criterion to create the $\mathcal{SS}^{init}$ spaces is problem dependent. Our proposed criterion for the NGPO is based on the partition of the search space $\mathcal{SS}$ into considerable smaller sets $\mathcal{SS}_i^{init}$ to speed up the search. This criterion will be introduced in Section 5.3.2.

5.2.1.2 Phase I: Search

Each $\mathcal{A}g_i$, including $\mathcal{A}g_M$, initializes its population, and performs the search until the termination condition $\mathcal{G}$ becomes TRUE after n_{PHASEI}-number of transitions. During PHASE I the positions of the particles of $\mathcal{A}g_i$ remain limited to the variability range imposed by $\mathcal{SS}_i^{init}$.

5.2.1.3 Phase I: Cooperation

Note that communication may not be necessary between the search agents during PHASE I other than for the assignment of the search instances and the collection of the results, as will be explained in Section 5.3.2.

5.2.1.4 Phase I: Termination

At the end of the search in PHASE I each search agent $\mathcal{A}g_i$ sends its resulting $\mathcal{PF}_{temp_i}$ to $\mathcal{A}g_M$. $\mathcal{A}g_M$ receives and merges the preliminary Pareto fronts provided by all search agents and produces a single set of non-dominated solutions, denoted as $\mathcal{PF}^{init}$. $\mathcal{PF}^{init}$ is then used by the $\mathcal{MASS}$ to initialize the particles in PHASE II.

In Summary, the objective of PHASE I in our proposed approach is to avoid sending the particles of the swarm to perform a blind search in the potentially immense $\mathcal{SS}$ space. The difficult part is to find an appropriate way to divide the given Ins into $n_{\mathcal{MASS}}$ instances, one for each search agent in the $\mathcal{MASS}$, to perform the search faster. An option we found suitable was to divide the search space using knowledge particular to the application, as will be shown in Section 5.3 in which details of the NGPO problem are provided.

5.2.2 Phase II

During PHASE II the entire team of search agents will explore the search space looking for $\mathcal{PF}^*$, the Pareto front that contains the set of optimal solutions. To do so, the $\mathcal{MASS}$ applies the distributed set-based PSO search system proposed in Section 4.2. The criterion to create $\mathcal{SS}^{init}$ in PHASE II is based on the use of $\mathcal{PF}^{init}$ as starting point for the search. The reason to do this is because the Pareto optimal set associated with $\mathcal{PF}^{init}$ indicates, up to that point in time, the most promising area to continue the search.

5.2.2.1 Phase II: Initial Population

The master agent Ag_M takes the position vectors ($\vec{x}$) from the Pareto optimal set associated with $\mathcal{PF}^{init}$ to create $\mathcal{SS}^{init}$. Each search agent re-initializes the positions of its particles randomly with $\vec{x}^{init} \in \mathcal{SS}^{init}$, $\vec{v}^{init} = \vec{0}$, and

$$\vec{x}_{BO_i}^{init} = \begin{cases} \vec{x}_i^{init} & \text{iff } \vec{x}_i^{init} \lhd_{\text{MOO}} \vec{x}_{BO_i} \\ \vec{x}_{BO_i} & \text{else} \end{cases}$$

$\forall i = 1, 2, ..., n_{swarm}$, for all search agents in the $\mathcal{MASS}$. The variability range of each decision variable dictated by $\mathcal{SS}^{init}$ is the variability range of the position vectors of the non-dominated solutions resulted from PHASE I. The operator $\lhd_{\text{MOO}}$ is a comparison criterion for MOO and will be explained in Section 5.3.1.

5.2.2.2 Phase II: Search

Note that if one or more search agents were assigned to bad search areas during PHASE I, they will be rescued in PHASE II. At this point in time each search agent $\mathcal{A}g_i$ is working on its search instance and updating its own $\mathcal{PF}_{temp_i}$.

5.2.2.3 Phase II: Cooperation

As in the single-objective case, the search agents communicate periodically with $\mathcal{A}g_M$ whenever the communication condition becomes TRUE after certain number of transitions to exchange specific pieces of information.

In early stages of PHASE II the $\mathcal{PF}_{temp_i}$ sets contain a small number of elements. At this time the search agents can obtain some benefit from learning where other swarms have found good solutions. For that reason, every time the communication condition $\mathcal{G}_{comm_{sol_i^+}}$ becomes TRUE after $n_{sol_i^+}$-number of transitions, each search agent $\mathcal{A}g_i$ makes use of its mes_i-function to provide k_1 solutions. Each search agent randomly selects these k_1 solutions from the $\vec{x}_{BO}$s of its swarm. All these solutions provided by the search agents are merged by $\mathcal{A}g_M$ eliminating those that become dominated. k_2 solution vectors are then selected out of the remaining solutions[15]. The k_2 solutions are used to randomly replace k_2 current solutions in each $\mathcal{A}g_i$'s state, essentially moving the position of these particles to hopefully better areas of the search space. The velocity of

[15] k_2 is the maximum number of positions that will be replaced. If the number of the remaining non-dominated solutions is smaller than k_2 only those remaining are replaced.

the reallocated particles remains the same, and the particle's best ancestor is updated only if the relocated position is better.

As the search progresses the number of elements in the sets $\mathcal{PF}_{temp_i}$ grows and may even reach its maximum capacity. At this time these sets are the best reference to guide the search. For this reason every time the communication condition $\mathcal{G}_{comm_{k_i^+}}$ becomes TRUE, each search agent $\mathcal{A}g_i$ makes use of its mes_i-function to provide its current $\mathcal{PF}_{temp_i}$. $\mathcal{G}_{comm_{k_i^+}}$ is set TRUE every $n_{k_i^+}$-number of transitions. $\mathcal{A}g_M$ merges the $\mathcal{PF}_{temp_i}$s found by the search agents and discards all solution vectors that become dominated. The task of $\mathcal{A}g_M$ is to obtain a $\mathcal{PF}_{temp}$ with maximum capacity of k_3 solution vectors, and provide it to all the search agents in the $\mathcal{MASS}$. If the number of solution vectors in $\mathcal{PF}_{temp}$ is larger than k_3, $\mathcal{A}g_M$ starts deleting solution vectors from the hypercubes with major density until the k_3 maximum quota is reached.

The type of communication exchange is controlled by linearly increasing $n_{sol_i^+}$ and linearly decreasing $n_{k_i^+}$ over time.

5.2.2.4 Phase II: Termination

Phase II ends when the termination condition $\mathcal{G}$ becomes TRUE after n_{PHASEII}-number of transitions.

With this 2-phase approach for MOO we expect the $\mathcal{MASS}$ be able to find a solution to a given instance of the search faster than the approach used for single-objective

optimization which is basically PHASE II only. Results of the experiments reported in Chapter 7, with and without PHASE I, demonstrate that the algorithmic design aspects of IOPO for MOO decreases the computational cost and makes it efficient to solve the complex MOO of NGPO problem.

5.3 Extending IOPO to Multi-Objective Optimization of NGPO

As explained in Section 4.3.1, the NGPO problem can be stated as a multi-objective optimization problem with conflicting objectives such as minimizing fuel consumption while maximizing throughput and also maximizing (or minimizing) the linepack in the pipeline system. In this thesis we studied and solved the 2-objective NGPO optimization problem for fuel minimization and throughput maximization.

From an operations point of view, studying the MOO case for the NGPO problem makes sense not only to be able to find the optimal vector $\vec{x}^*$ to operate the pipeline network at a specific throughput with minimum fuel consumption, but also to know other Pareto optimal vectors in the vicinity of $\vec{x}^*$ that may contain slightly different operational settings. For example, two non-dominated solutions located very close in the objective space may imply the use of different CSs in the decision variable space. Having a few sets of operational settings available may be helpful in case a CS has to be OFFLINE for reasons such as maintenance or upgrade of the equipment. Having additional operating settings provides flexibility to operate the network close to optimal conditions. Also, the full Pareto front and the Pareto optimal set provide the complete

spectrum of variation of the operating settings of the pipeline network if operated in optimal conditions.

As mentioned before, sending the swarm to perform a blind search is simply not an option because the search space is huge and it will take too much time to find useful solutions. But operation of a pipeline network requires the solution in a timely manner. With this time restriction in mind we proposed the 2-phase approach introduced in Section 5.2 aimed to reduce the time needed to find the solution. In this section we explain how the proposed 2-phase approach is applied to solve the MOO of the NGPO problem. But first we need to define the criteria to compare solution vectors, an important definition that will be used later.

5.3.1 Criteria to Compare Solutions

In Section 4.3.2.2 we proposed a **lexicographical combination of orderings** to compare two position vectors based on the triple $(\mathcal{HC}(\vec{x}), \Omega(\vec{x}), f_1(\vec{x}))$, where $\mathcal{HC}$ indicates the hydraulic correctness of the solution vector $\vec{x}$, Ω accounts for all the violations caused by the set of operational settings encoded in $\vec{x}$ if implemented in the pipeline network, and f_1 indicates the fuel consumed by the CSs.

For the MOO case we also use a lexicographical combination of orderings to compare position vectors based on the triple $(\mathcal{HC}(\vec{x}), \Omega(\vec{x}), \vec{f}(\vec{x}))$. Note that this time the criterion includes a vector of objective functions $\vec{f}(\vec{x})$ instead of just a single objective. Now, the position of particle 1 is considered better than the position of particle 2

$(\vec{x}_1 \lhd_{\text{MOO}} \vec{x}_2)$, if either $\mathcal{HC}(\vec{x}_1) > \mathcal{HC}(\vec{x}_2)$ or, if $\mathcal{HC}(\vec{x}_1) = \mathcal{HC}(\vec{x}_2)$ and $\Omega(\vec{x}_1) < \Omega(\vec{x}_2)$ or, if $\mathcal{HC}(\vec{x}_1) = \mathcal{HC}(\vec{x}_2)$ and $\Omega(\vec{x}_1) = \Omega(\vec{x}_2)$ and $\vec{f}(\vec{x}_1) \prec \vec{f}(\vec{x}_2)$. The symbol $\prec$ indicates dominance as introduced in Definition 2.1 in Section 2.1. In other words, hydraulically correct solutions are always better than incorrect ones, solutions with less constraint violations are better than solutions with more violations, and valid solutions are compared based on the Pareto dominance criterion which considers the value of all the objective functions simultaneously. Now, if we go back to the set of extension rules $\mathcal{E}xt$ for the MOO case introduced in Section 5.1.1, we can see that a particle's best solution ($\vec{x}_{BO}$) will only be updated if it becomes dominated. In other words, a newly created non-dominated solution not dominating the current $\vec{x}_{BO}$ has no effect in this particular value.

5.3.2 Phase I

We identified **knowledge about the NGPO problem** and used it in the MOO case to divide $\mathcal{SS}$ into smaller $\mathcal{SS}^{init}$ sets. It is important to mention that selection of this knowledge to improve the search is critical, and not a trivial task, especially for a MOO problem with conflicting objectives because conflict can arise between the knowledge that optimizes one of the objectives but conflicts with the knowledge used to optimize the other objective.

In our case in which one objective is to maximize throughput, knowledge about NGPO will direct the search towards the states in which most CSs are ONLINE, and

perhaps working at maximum capacity –within the physical limits– in order to move as much gas as possible and maximize the throughput at the control node of interest. However, because the second objective is to minimize the fuel consumed by compressor units at the CSs in the pipeline network, then knowledge particular to this objective will try to use the smallest possible number of CSs to reduce the gas consumption, which contradicts the solution to the former objective. Patterns of behavior like these could not be easily applied without affecting one or the other objective.

We observed that among the decision variables used to define the NGPO problem there is one that is of particular interest. We will refer to this decision variable as Pr, where $Pr \in [Pr_{min}, Pr_{max}]$. Pr is one of the continuous decision variables in $\vec{x}$, the vector of operational settings of the pipeline network, and is associated to the amount of gas received in the pipeline network[16].

This knowledge about NGPO was used to partition the decision variable space and create $\mathcal{SS}^{init}$. Based on the meaning of Pr, $\mathcal{SS}$ is reduced to a number of smaller $\mathcal{SS}^{init}$ sets by restricting the value that Pr can take. Each $\mathcal{SS}^{init}$ is practically equal to $\mathcal{SS}$ except for the range allowed for Pr. The range of Pr is partitioned into Pr_1, Pr_2, ...,$Pr_{n_{Pr}}$ segments covering the complete range of Pr. With this condition, each instance corresponds to a specific area of the search space.

The number of segments (n_{Pr}) is determined depending on the size and complexity

[16]All the pipeline networks studied in this thesis are modeled by the hydraulic simulator as a closed system, i.e. the amount of gas that comes into the pipeline network should be equal to the amount of gas delivered plus the gas consumed by the compressor units.

of the optimization problem, i.e. the pipeline network. For small pipeline networks, depending on the number of processing units available, a number of segments $n_{Pr} = n_{\mathcal{MASS}}$ could work well. In other words, $\mathcal{SS}$ is partitioned into $n_{\mathcal{MASS}}$ smaller sets and the search agents initialize their population according to Section 5.2.1.1, with $\mathcal{SS}^{init}$ defined as described in the previous paragraph. Each of the $n_{\mathcal{MASS}}$ search agents works on the received instance, does the search, and at the end reports its $\mathcal{PF}_{temp_i}$ to $\mathcal{A}g_M$.

For larger and more complex pipeline networks the performance of the search can be improved by partitioning $\mathcal{SS}$ into more pieces than before[17], i.e. with $n_{Pr} > n_{\mathcal{MASS}}$[18]. This means that one or more search agents will be assigned more than one Ins_i. If a search agent is assigned more than one instance it will first work on one instance, then continue with another one, and so on, until the search agent processes all the instances it received. This may appear a very long process but it is not for two reasons.

First, we use again the NGPO knowledge described in Section 4.3.2 which reveals that solutions to the NGPO problem that are close in the objective space are also close in the decision variable space[19]. We use this knowledge in the following way. The solution vectors associated to $\mathcal{PF}_{temp_i}$ for the segment Pr_i are used as starting point for the search in the adjacent segment Pr_j, where $Pr_1, Pr_2, ..., Pr_i, Prj, ..., Pr_{n_{Pr}}$, a strategy that definitely speeds up the search.

[17]This is similar to what Suttner did in [Suttner 1995].

[18]Preferably with n_{Pr} being a multiple of $n_{\mathcal{MASS}}$.

[19]As mentioned earlier, it is important to note that this behavior is followed only by those decision variables that are continuous ($x_i^c \in [P_{min}, P_{max}]$, $i = 1, 2, ..., n_c$). Discrete variables not necessarily behave like this, their value may change abruptly which still gives complexity to the search problem.

Second, knowledge about NGPO indicates that the number of existing valid solutions to operate the pipeline network at low throughput levels is larger than the number of valid solutions existing to operate the pipeline network at higher throughput levels. The reason is because the pipeline network has more flexibility to satisfy the transportation requirements at low throughput levels, and this has an impact on computation time. We decided to organize the search in the $\mathcal{MASS}$ as follows. $\mathcal{SS}$ is divided into the smaller sets $\mathcal{SS}_1^{init}$, $\mathcal{SS}_2^{init}$,..., $\mathcal{SS}_{n_{Pr}}^{init}$, where n_{Pr} is a number of partitions that convert the original search space into smaller, tractable search spaces. The selection of n_{Pr} is a parameter of our system ($n_{Pr} > n_{\mathcal{MASS}}$) and requires some tuning.

Now, instead of assigning one search agent to a particular $\mathcal{SS}_i^{init}$ ($i = 1, 2, ..., n_{Pr}$), all search agents in the $\mathcal{MASS}$ are assigned to Ins_1 to search for $\mathcal{PF}_{temp1}$. The search agents initialize their population following the approach introduced in Section 5.2.1.1. They perform the search within $\mathcal{SS}_1^{init}$ and communicate as described in Section 5.2.2.3. The search in Ins_1 ends after a determined number of transitions. The resulting solution vectors associated to $\mathcal{PF}_{temp1}$ are used to initialize the positions of the particles of the search agents as described in Section 5.2.2.1 to start searching in Ins_2. This process is repeated until the n_{Pr} instances are completed. $\mathcal{A}g_M$ does the union of the n_{Pr} nondominated sets to generate $\mathcal{PF}^{init}$. This newly generated set is used to initialize the search in PHASE II.

5.3.3 Phase II

As mentioned earlier, PHASE II is almost identical to the distributed set-based PSO search system introduced for single-objective optimization in Chapter 4. However, because MOO is a much more complex problem to solve we added PHASE I to improve the selection of $\mathcal{SS}^{init}$.

The end result of PHASE I is a set of non-dominated solution vectors ($\mathcal{PF}^{init}$) found by the $\mathcal{MASS}$. The position vectors associated to those non-dominated solutions are used as reference to create $\mathcal{SS}^{init}$.

In PHASE II, all search agents in the $\mathcal{MASS}$ re-initialize the position of their particles randomly with $\vec{x}^{init} \in \mathcal{SS}^{init}$, as explained in Section 5.2.2.1. Note that none of the components of the newly created vectors $\vec{x}_i^{init}$ should take a value outside of the physical constraints imposed by the original $\mathcal{SS}$.

As for the single-objective case, the position of the particles remains limited to the variability range imposed by $\mathcal{SS}^{init}$ during n_{PHASEII_0} number of transitions and then the range is reset to the original, and wider, range defined by $\mathcal{SS}$. As mentioned earlier, early experiments not documented in this thesis showed that opening the range of the decision variables is important to have the opportunity to rescue the swarm in case it becomes trapped in a bad area of the search space and also to have the opportunity to search for better solutions that might be outside of the space delimited by $\mathcal{SS}^{init}$.

At this point in time all the search agents in the $\mathcal{MASS}$ are performing search and evaluating solution vectors using their own instance of the hydraulic simulator

and updating their velocities and positions according to the set of extension rules $\mathcal{E}xt$ introduced in Section 5.1.1. Each search agent $\mathcal{A}g_i$ keeps updating the record of the best solutions found so far in its $\mathcal{PF}_{temp_i}$.

The search agents in the $\mathcal{MASS}$ communicate as explained in Section 5.2.2.3. The search ends when the termination condition $\mathcal{G}$ is set TRUE. At the end $\mathcal{A}g_M$ presents $\mathcal{PF}_{temp}$, which represents the best solutions found by the $\mathcal{MASS}$ to the given search problem.

6 Related Work

This chapter summarizes relevant work related to the various topics involved in this thesis and is organized as follows. We start with Section 6.1 presenting a summary of previous approaches aimed at the optimization of NGPO. This particular section also introduces two state-of-the-art methods that we use in the experiments reported in Chapter 7 to compare them against our IOPO approach. Section 6.2 lists the most relevant distribution approaches and Section 6.3 describes relevant PSO work done for the particular case of multi-objective optimization.

6.1 Other Approaches for Optimization of Pipeline Operations

In some situations the operators of the compressor stations maintain the desired throughputs by shutting down or controlling individual compressor units' speeds based on experience, generally a trial and error process without any guarantee of optimality [Krishnaswami et al. 2004]. In better cases, network operators run their best guesses in hydraulic simulators of the pipeline network to determine the reaction of the system

for a given set of operating parameters. Gas controllers are required to scan the pipeline system for upsets and emergencies while monitoring operating valves and compressors as often as necessary for safe and efficient transmission of the scheduled gas volumes from their operations control centre [Kerkhof et al. 2002]. Reality is that the number of variables that gas controllers, operators and planners have to consider simultaneously is very large and minimization of fuel consumption is an objective that cannot be expected from human calculations.

An enormous amount of work has been invested in the past to develop an automatic method to optimize pipeline operations. In the following paragraphs we list some of these approaches, noting that some of them were applied to the transportation of natural gas and others for fluids such as oil and water. As stated in earlier sections of this thesis, the principles to operate and perhaps optimize the operation of this kind of transportation systems are similar -to a certain extent- and that is the reason to include these references as related work.

The earliest work on developing optimization algorithms for natural gas pipeline networks can be tracked back to 1968 ([Wong and Larson 1968] in [Wu 1998]) when **dynamic programming** techniques were used to solve optimization problems for simple pipeline networks.

Early research on NGPO optimization focused on techniques such as enumeration, i.e. the evaluation of all possible solutions. **Enumeration** severely limits the size and complexity of the pipeline network systems that can be solved. It can only be applied

to simple and small pipeline networks; otherwise it becomes a prohibitively expensive and definitely a time-consuming approach. In [Veloso et al. 2004] a spreadsheet-based computational tool was used to reduce the energy consumption at each pumping station in oil pipelines. The methodology was based on the construction of a database relating the pumping power consumption for all possible pumping arrangements and viable flow rate ranges of the pipeline network under consideration. The cost of each pumping arrangement was calculated by running each combination using the Stoner Pipeline Simulator V9.31 as hydraulic simulator. After running all the possible pumping arrangements the data is exported to a worksheet avoiding the need of running the simulator again in the future. The spreadsheet was then used to select the minimum pumping cost for each operation point as needed. This method was used for pipeline networks consisting of only 3 pumping stations, where each station was equipped with either 3 or 4 pumps respectively.

Enumeration and dynamic programming were combined in [Bolkan 1991] for optimal design and operation of a pipeline network. All pump combinations at each pumping station were arranged in increasing discharge pressure order, then selecting the pump combination just sufficient to pump the fluid at a required pressure. A solution for the entire network is found by using Bellman's principle of optimality. Numerical examples are reported for 15 pumping stations with 3 pump units each, and 10 possible suction pressures.

Other approaches aim to simplify the optimization problem through the linearization of the objective function and constraints. [Goslinga et al. 1994] proposed an exact first order linearization of the constraints by first order Taylor's series expansion and the linear programming problem is solved using Simplex.

[Krishnaswami et al. 2004] used **sequential unconstrained optimization** with an exterior penalty function to minimize the total fuel consumption while maintaining a desired throughput. The speeds of the centrifugal compressors (with similar or dissimilar units) were selected as the decision variables and the solution was constrained by minimum and maximum speed limits, and by minimum mass flow rate at the compressor station. The work was designed for optimization of only one compressor station. The experiments reported results for a single compressor station with up to 10 compressor units. The authors claim the methodology can be extended beyond the compressor station level to the network level but no results were presented.

Other approaches are aimed at the reduction of the dimension of the optimization problem. [Wu 1998] used a combination of graph theory and non linear functional analysis to determine linear supersets of the feasible solutions and convex lower bounding for the cost function. Based on those results a **network decomposition method** was introduced to reduce the size and difficulty of the problem. Among the numerical experiments provided in his work, the most complex problem instance was a pipeline network with 8 compressor stations with identical centrifugal compressor units in parallel and did not include other pipeline elements such as block valves, control valves or

regulators. The research work continued and later on [Rios-Mercado et al. 2002] did a theoretical study of the properties of a gas pipeline network and exploited those properties to reduce the problem dimension. The authors proposed a **network reduction method** to reduce the number of independent variables and non linear equality constraints. The research work reported important theoretical findings but did not include any numerical examples. Further work using reduced graphs combined with dynamic programming with a given pressure discretization was presented in [Rios-Mercado 2002] considering only the case of identical units in series for isothermal models. Unfortunately no experimental results were reported.

None of the traditional approaches were able to optimize realistic pipeline networks mostly because of the complexity imposed by the pipeline network modeling. Nevertheless, with advances in computing power and biologically-inspired search systems more research groups have extended their studies to explore optimization techniques applied to problems in transmission networks of larger size and increased complexity, in other words, for more realistic scenarios.

Genetic algorithms, and more recently particle swarm optimization and ant colony optimization, have been used to optimize the design of water distribution systems, for the most part to determine design parameters such as pipe diameters prior to the construction of the pipeline networks [Simpson et al. 1994, Dandy et al. 1996, Zhang 1999, Maier et al. 2003, Jung and Karney 2004, Eusuff 2004]. In the area of optimization of pipeline operations, genetic algorithms and simulated annealing have been used for

natural gas pipeline systems as early as in 1983.

[Goldberg 1983], [Goldberg 1987a], and [Goldberg 1987b] proposed a decision-making system to operate and control gas pipeline networks in normal summer and winter conditions as well as to learn how to detect leaks in the system. A learning classifier system was developed to create, evaluate and exploit string rules to control a simulated pipeline system. New rules are created using genetic algorithms by reproducing, crossing and mutating rules contained in the current rule set. The proposed approach was tested for steady state using a simple serial system composed by 10 compressor units and 10 pieces of pipe. In other words, this scenario could be interpreted as 10 compressor stations with only one compressor unit inside. This configuration has significantly less complexity compared to realistic pipeline networks –like the ones studied in this thesis– in which compressor stations have several compressor units inside, i.e. more settings of operating options from where pipeline operators have to choose from to reduce the transportation cost. Goldberg's work also included the transient case in a single line, i.e. with one compressor and one pipe.

[Wright et al. 1998] applied **simulated annealing** as technique for finding the optimum configuration and power settings for multiple compressors arranged in series or parallel. The objective of the work was first aimed for fuel optimization at the compressor station level by searching for the best arrangement of the compressor units at the station rather than aiming for the optimization of operation of the complete pipeline network. The compressor stations were assumed to be in series with a fixed

pressure drop between any two adjacent compressor stations from the discharge of the upstream station to the suction of the downstream station. Although results are reported for pipeline networks as complex as 25 compressor stations with 10 compressor units each, it is not clear if many assumptions were considered to simplify the problem and report processing times in the order of 6.5 minutes.

[Botros et al. 2004] applied **genetic algorithms** to optimize large gas pipeline networks. The model of the pipeline network included several elements such as line crossovers, multi-unit compressor stations, block and control valves. Results on sub-networks up to the size of 25 compressor stations revealed that the main limiting factor of this approach was the computation effort caused by the great number of evaluations necessary to find a satisfactory solution. This approach used a binary representation of the decision variables and, as indicated by [Michalewicz 1996], the binary representation has some drawbacks when applied to multidimensional, high precision constrained optimization problems. In [Botros et al. 2004] the binary vector used to represent the decision variables of a small pipeline network consisting of only 8 compressor stations had a length of 99 bits implying a search space of about 6.33×10^{29}. Unfortunately, for a larger and more realistic pipeline sub-network represented with a binary vector of 260 bits the space grows dramatically to 1.85×10^{78} cases. [Zhang 1999] also applied genetic algorithms for pipeline network design and reported a large decision space of 10^{21} generated by strings of 84 bits of length. The research for large and complex pipeline sub-networks was continued in [Stoffregen et al. 2005] and

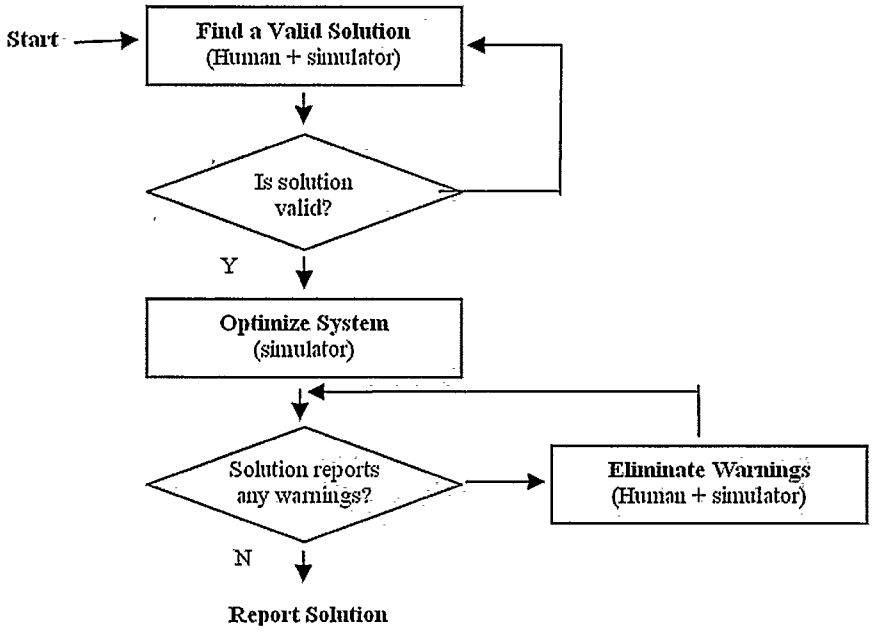
[Botros et al. 2006] where surrogate methods were introduced aimed to reduce the time consumed during the evaluations of the solution vectors. However, the prolonged runtimes needed to optimize realistic pipeline sub-networks continued to be the main drawback of this approach eliminating any possibility to be applicable to the day-by-day operations. Results in Section 7.2 show the comparison between the performance of this approach (called MOGA) and our proposed approach (called IOPO) showing that our approach satisfies the essential time constraints required by pipeline operators.

The Commercial Approach. One of the problems of existing software tools for fuel optimization is that they are unable to handle devices which have only two states such as block valves that can be FULLYOPENED and FULLYCLOSED. Additionally, software tools are limited to a specific number of decision variables which automatically restrict the area of the pipeline network that can be modeled and optimized. As a consequence at TransCanada PipeLines, the established practice involves the use of in-house developed simulation software to analyze pipeline operations and determine the operating configuration that optimizes the fuel consumption on the pipeline system. This approach requires the user to provide a valid initial solution. The process of finding improved operating configurations is an iterative one. The user makes adjustments to the input and repeats the simulation several times with the adjusted input until successively improved solutions are found. A major shortcoming of the established method is the significant amount of human interaction and intervention, and therefore the significant time and effort required to successively iterate to a solution.

Due to these problems, TransCanada PipeLines has been actively researching methods and evaluating commercial products to improve upon the current practice. In fact, TransCanada PipeLines supported the research presented in [Botros et al. 2004] and [Botros et al. 2006] which developed the MOGA method that used Genetic Algorithms to solve the problem fully and automatically. However, as already stated, this research was not able to provide the run-time performance that TransCanada PipeLines needs. The research in [Botros et al. 2004] and [Botros et al. 2006] used hydraulic simulation software developed in-house at TransCanada PipeLines. For this thesis work we used the same hydraulic simulator.

In addition to the MOGA research activities, TransCanada PipeLines has been evaluating the suitability of commercial optimization software for the purpose of optimizing fuel usage on the pipeline system and reduce emissions. For the purpose of this thesis the commercial software is referred to as the **commercial method**. Similar to the in-house developed method, shortcomings have been encountered with the commercial method of fuel optimization. This method also requires the user to find a valid and feasible solution as a starting point for the optimization to be able to start and also requires a significant amount of human intervention to successively iterate to a solution. This iterative process, illustrated in Figure 6.1, makes it difficult to predict its duration and it is clear that the experience of the user has strong influence on the time needed to obtain the results. In addition, some pipeline components such as block valves can be modeled but not handled as decision variables by the optimiza-

Figure 6.1: Current Approach: Human-Simulator-Optimizer Interaction.



tion method. In Chapter 7 we use this commercial method as comparison point in our experiments to reflect the state-of-the-art in the field. Neither the commercial method nor the MOGA optimization methods have been developed to take advantage of distributed computing.

6.2 Approaches for Distributing/Parallelizing Evaluations

Real world optimization problems are characterized by 1) requiring a large number of evaluations of solutions vectors and/or 2) evaluations that are computationally expensive, i.e. that take too much time. There are various approaches to tackle these kinds of problems. Some methods are aimed to reduce the computational cost of the evalu-

ation by using approximation functions as the surrogate methods used in [Stoffregen et al. 2005] and [Botros et al. 2006]. Other methods opt for distributing the evaluations in various processor units. This section summarizes work related to distributed approaches.

Evolutionary algorithms have become *the* method at hand for multi-objective optimization (MOO) problems that are too complex to be solved by exact methods such as linear programming and gradient search due to the inherent parallelism and their capability to exploit similarities of solutions [Zitzler et al. 2000]. Many multi-objective evolutionary algorithms (MOEAs) are successfully applied to real-world optimization problems but achieving better efficiency remains a goal. The attractive characteristics of parallel MOEAs include concurrent search for multiple solutions, ease of parallelizing serial MOEAs, reduction of wall clock execution time, and achieving better overall effectiveness [VanVeldhuizen et al. 2003]. For a thorough discussion of evolutionary algorithms for MOO the reader is referred to [Coello 2006].

In single-objective and multi-objective optimization, expensive objective function evaluations (in terms of CPU time), are often completed in less wall clock time by decomposing the computational load across two or more processors. One approach is to utilize parallel function decomposition computational techniques. Another approach is to spatially decompose the population across a given set of processors. In general, distributed multi-objective evolutionary algorithms are classified into four broad streams: master-slave, island (or coarse-grained), diffusion (or fine-grained), and hybrid, each of

which may be implemented in either synchronized or non-synchronized way. Although there are other classifications of distributed search in literature we present [VanVeldhuizen et al. 2003]'s classification since it is rather simple.

In the **master-slave** scheme objective function evaluations are distributed among several processors called slaves. It can be that all slaves evaluate all objective functions or also that different objectives (or subsets of objectives) are assigned to different slaves. Although the master processor could also be used for objective function evaluations in general it is dedicated to execute evolutionary operators and other overhead functions. This scheme helps to reduce the runtime when the problem involves reasonably intricate and time consuming objective function evaluations in a proportion that the communication time between processors is insignificant.

In the **island** scheme the population is divided among the available processors and each processor is considered an island. The individuals in each island perform the search on their own and sporadically exchange (migrate) individuals with other islands. The frequency of the communication, the number of individuals to exchange, and the criterion to select and replace individuals are all parameters to set. Each of the islands can have the same or different operators or search algorithms than the rest of the islands.

In the **diffusion** scheme the number of individuals per processor is very small and the processors are assigned either to overlapping or dynamically changing neighborhoods. The evolutionary operators are applied within each neighborhood, and

whenever improved solutions are found they are slowly diffused to the rest of the neighborhoods.

One kind of **hierarchical hybrid** scheme is a tree or graph search structure designed to find better algorithmic structures for a given problem. The bottom nodes contain different MOEAs with specific parameters. The next level of nodes evaluates the performance of the bottom nodes and adjusts the parameters to improve the MOEAs performance, and this process is continued until it reaches the top node.

6.3 PSO for Multi-Objective Optimization Problems

Several applications of PSO to MOO problems have been proposed before, for example [Hu and Eberhart 2002] and [Toscano-Pulido and Coello 2004]. A comprehensive review of MOO PSO can be found in [Reyes-Sierra and Coello 2006]. This section introduces PSO applications relevant for this thesis.

[Xiao-hua et al. 2005] introduced an approach for MOO based on the **Agent-Environment-Rules** initially introduced in [Liu et al. 2002]. The authors modified the general velocity equation to include the influence not only from personal and global experience but also local experience from the neighborhood. The neighborhoods are defined as in a lattice topology where all particles reside in an agent lattice environment. The particles can either compete or cooperate with its neighbors to acquire more resources which is a measure of fitness. A clonal selection operator is used to accelerate

the search. The efficiency of this approach was tested with benchmark problems from [Zitzler and Thiele 1999].

The ***covering*** **MOPSO** proposed in [Mostaghim and Teich 2004] consists of two phases: the initial run and the covering phase. In the initial run a MOPSO (multi-objective PSO) algorithm searches the whole search space for a good approximation of the Pareto front with restricted archive size. In the covering phase, the particles of the population are divided into sub-swarms and each sub-swarm is assigned to a non-dominated solution obtained as result of the initial run. The objective of the sub-swarms during the covering phase is to fill the gaps between the non-dominated solutions obtained in the initial run; in this phase there is no limit in the size of the archive. The *covering* MOPSO is based in the strong assumption that the initial run will provide relatively well distributed solutions very close to the Pareto-optimal front. The method was tested using four test functions from [Deb 2001], and also using an antenna design study which included only 12 decision variables. The total number of evaluations in the experiments (including the initial run and the covering phase) varied from 140,000 to 1.2 million. This method can not be applied to solve problems like the optimization of NGPO because its success relies in a huge number of function evaluations. The time restrictions imposed to the NGPO optimization problem require a search algorithm capable of solving complex pipeline sub-networks using a limited number of function evaluations because they are very time consuming. On the other hand, it is not clear if the *covering* MOPSO method is efficient for

high dimensional problems because it was tested with optimization problems with dimension no larger than 30 decision variables. A better approach, robust and scalable, is definitely needed to solve problems such as the NGPO which are characterized not only by a high dimensionality but also by the mix of continuous and discrete variables. Section 7.2 shows the capability of our method to tackle these kinds of problems.

In [Janson and Merkle 2005] the authors proposed **ClustMPSO**, an hybrid approach based on the combination of K-means clustering and PSO to solve MOO problems. The clustering is applied to the particle's best positions resulting in K clusters or swarms. Each of these swarms has its own non-dominated front. The union of all non-dominated fronts is the total non-dominated front. Particles within these non-dominated fronts can be dominated by particles within other swarms. The particle's best position is updated if it becomes dominated or, with certain probability is replaced by a dominated particle. Each particle randomly selects the swarm's best particle from the non-dominated front of the swarm to which it belongs. The particle updates this value periodically, after a certain number of transitions, or earlier if its swarm's best position becomes dominated. In other words, particles in a swarm most likely have different leaders. If all particles of a swarm are dominated by the total non-dominated front for a certain number of iterations the swarm is relocated. ClustMPSO was tested on an artificial multi-objective optimization problem with 3 real valued decision variables and a disconnected-continuous front. It was also tested on two instances of the molecular docking problem from biochemistry using 500,000 evaluations. This ap-

proach is not appropriate for complex and highly dimensional optimization problems such as the NGPO in which the search space is immense and, contrary to the problems treated in [Janson and Merkle 2005], the particles of the swarm will take too much time to find a non-dominated solution, if they eventually find one. ClustMPSO assumes that during the initial phase the swarm will be able to find a set of non-dominated solutions. This assumption cannot be made for the NGPO case, which is particularly complex and a single swarm will take too much time (time$\rightarrow \infty$) to find an acceptable set of preliminary non-dominated solutions. This is demonstrated in the experiments reported in Section 7.4, in which we compare the result of using a practically blind search opposite to using our proposed limited search.

7 Experimental Evaluation

The experiments reported in this chapter were done with real instances of TransCanada PipeLines Ltd. network:

> "'TransCanada is a leader in the responsible development and reliable operation of North American energy infrastructure. Its network of more than 59,000 kilometers of pipeline taps into virtually all major gas supply basins in North America. TransCanada is one of the continent's largest providers of gas storage and related services with approximately 360 billion cubic feet of storage capacity. A growing independent power producer, TransCanada also owns, or has interests in, approximately 7,700 megawatts of power generation"'[20]

Optimization of pipeline operations is a very competitive research area as significant improvements in operations can generate considerable commercial advantage over competitors. Due to the proprietary nature of TransCanada PipeLines Ltd.'s data and the commercial sensitivity of the results the information revealed in this chapter, including the real cost of the solutions, is detailed only at a very high level.

This chapter is organized as follows. Section 7.1 describes the settings used for the experiments as well as the problem instances of the NGPO problem studied in this thesis. Section 7.2 demonstrates IOPO's performance against other approaches

[20] www.transcanada.com

for the optimization of NGPO. Results in Section 7.3 illustrate the effect of the Multi-Agent/Hardware platform aspect, and Section 7.4 shows the results of IOPO when applied to the MOO of the NGPO problem.

7.1 Settings of the Experiments

This section includes the settings for the experiments of this thesis. The first section describes the natural gas pipeline sub-networks used as problem instances. The second section provides details of the hardware used for the different experiments.

7.1.1 Problem Instances

Large pipeline networks are complex systems that often have to be broken down into more manageable parts to be operated (and perhaps optimized) independently. The nodes where these parts or sub-networks connect are called exchange nodes and their pressures and flows are set to specific values by human operators. For this thesis we use six problem instances from two different pipeline sub-networks within TransCanada PipeLines Ltd.'s pipeline system. These sub-networks are called SubNetwork 1 (SN1) and SubNetwork 2 (SN2).

SN1 is a small pipeline sub-network located in southern Alberta, Canada, which includes 6 compressor stations and one control valve. SN1 is illustrated in the shaded area in Figure 7.1, and the resulting 13 decision variables (n_c=7, n_d=6) necessary to describe this sub-network are listed in Table 7.1. On the other hand, SN2 is a

considerable larger sub-network which is illustrated in Figure 7.2. SN2 consists of 20 compressor stations, 8 control valves, 3 block valves, one control pressure at a meter station, and one proration ratio variable, resulting in a total of 53 decision variables (n_c=29, n_d=24) listed in Table 7.2, where

- P_{max} is the maximum allowable operating pressure (CSs and CVs) or the maximum contract pressure (CS, CV, meter station), whichever is less
- P_{min} is the minimum suction pressure (CS, CV, meter station)
- Pr_{max} is the maximum meter station proration factor
- Pr_{min} is the minimum meter station proration factor

SN1 and SN2 are actual divisions of the pipeline network that are optimized separately in TransCanada PipeLines Ltd's current practices. Naturally these sub-networks connect to other sub-networks respecting the boundary conditions. We considered three different scenarios of flow requirements for each of the sub-networks namely high flow (HF), medium flow (MF), and low flow (LF), for a total of six problem instances.

7.1.2 Hardware

The experiments with IOPO reported in Sections 7.2 and 7.4 were run on a 2 Dual Core Intel Xeon 5160 3.00 GHz workstation with 2GB of RAM running Windows XP as the operating system. Each search agent in IOPO was assigned to its own processing unit, with one of these processes also realizing the start, end, and master search agent. The

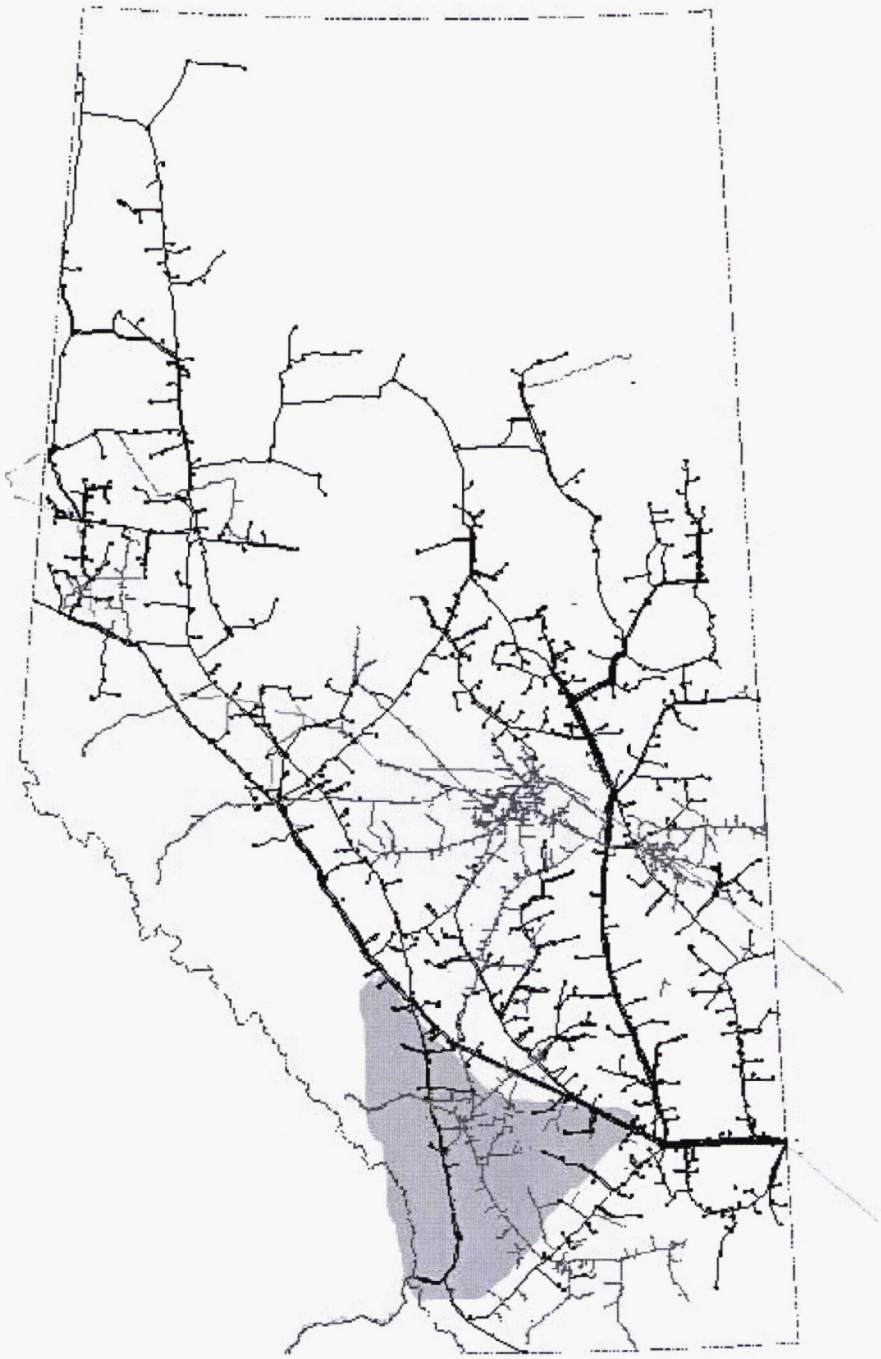


Figure 7.1: Natural gas pipeline SubNetwork 1 in Alberta, Canada.

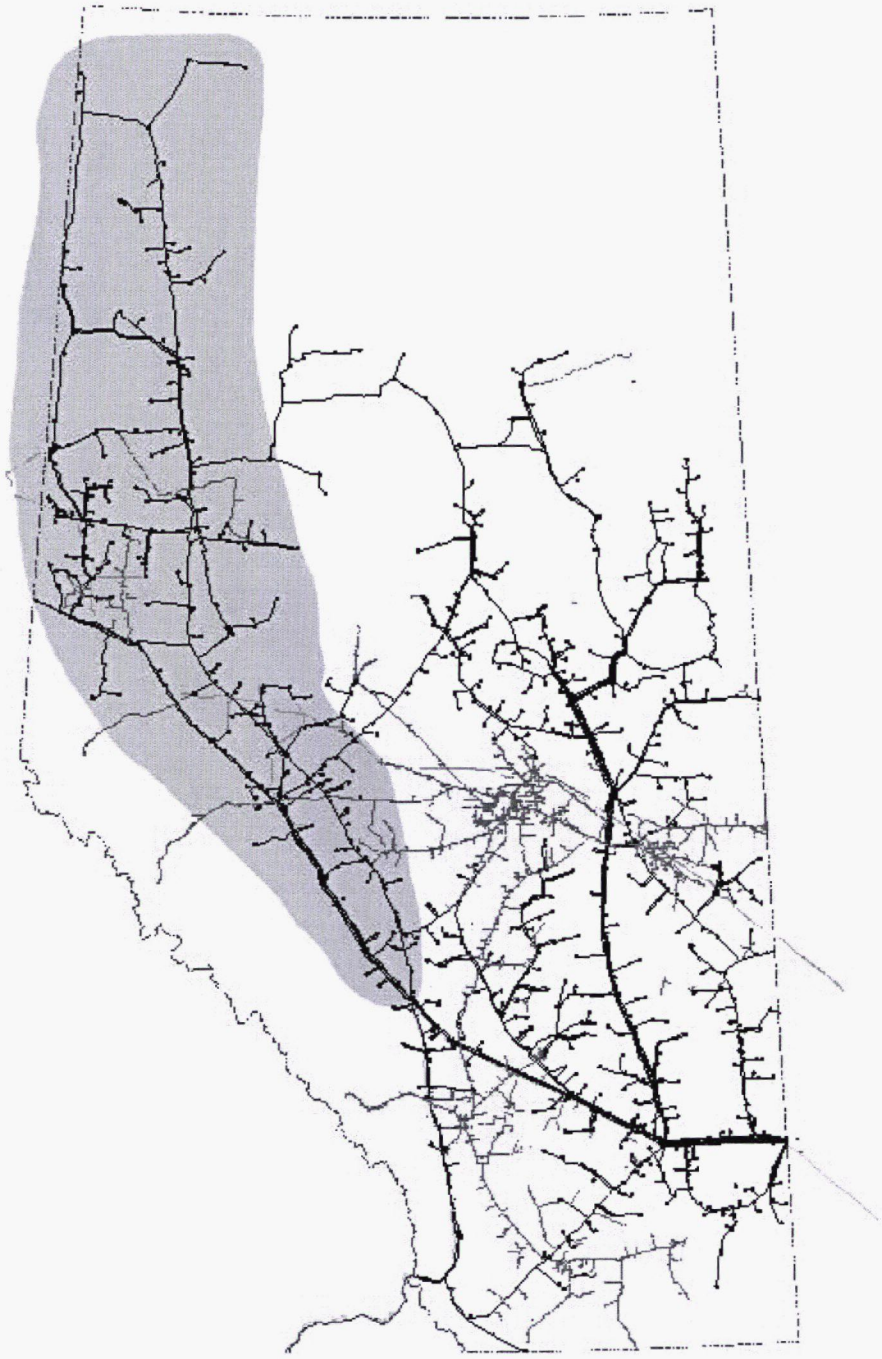


Figure 7.2: Natural gas pipeline SubNetwork 2 in Alberta, Canada.

Table 7.1: List of decision variables considered in SUBNETWORK 1

QUANTITY	TYPE OF DV	ALLOWED VALUES
6	CS pressure	$[P_{min}, P_{max}]$
6	CS status	{ONLINE, OFFLINE}
1	CV pressure	$[P_{min}, P_{max}]$

Table 7.2: List of decision variables considered in SUBNETWORK 2

QUANTITY	TYPE OF DV	ALLOWED VALUES
20	CS pressure	$[P_{min}, P_{max}]$
20	CS status	{ONLINE, OFFLINE}
8	CV pressure	$[P_{min}, P_{max}]$
3	BV status	{FULLYOPENED, FULLYCLOSED}
1	Meter station pressure	$[P_{min}, P_{max}]$
1	Proratio ratio	$\{Pr_{LF}, Pr_{MF}, Pr_{HF}\}$ or $[Pr_{min}, Pr_{max}]$

communication structure $\mathcal{K}om$ was done by having the mes_i send messages between the processes. IOPO is implemented in C.

The experiments in Section 7.3 were run using four Pentium III, 996 MHz, with 256 MB of RAM and a BayStack 450 at 100 Mbps.

7.2 Comparison of IOPO with Other Systems

The following series of experiments are aimed to demonstrate the performance of IOPO when applied to the problem of fuel minimization for TransCanada PipeLines Ltd's pipeline operations. IOPO is compared against MOGA and the *commercial method* introduced in Section 6.1. The experiments in this section are aimed at the minimization of the cost of transportation for a fixed throughput using steady state simulations of natural gas pipeline networks.

The IOPO parameters for the problem instances SN1/HF, SN1/MF, and SN1/LF are $n_{\mathcal{MASS}}$=4, n_{swarm}=10, n_{init}=15, n_{comm}=5, $\mathcal{G}$=50. The IOPO parameters for the three problem instances SN2/HF, SN2/MF, and SN2/LF are $n_{\mathcal{MASS}}$=4, n_{swarm}=10, n_{init}=30, n_{comm}=10, $\mathcal{G}$=100. The wall clock time of the commercial method includes the system's run time plus the estimated worst case interaction time for an experienced user which is 5 minutes for SN1 and 40 minutes for SN2. For IOPO, we report in both tables the average over 10 runs (due to the heavy use of random factors in the search). While MOGA also uses random factors, we only performed one run for each problem instance due to the very long runtimes for SN2.

As Table 7.3 shows, IOPO and the commercial method need approximately the same time to produce their results, both within the acceptable time requirements. Note that IOPO is fully automatic and does not require a user to provide a valid starting solution. It uses 4 search agents, one for each available core. MOGA's run time for the large sub-network (SN2) is far too long to meet TransCanada's time requirements to support pipeline operations on a day-to-day basis.

In order to compare the quality of the solution of the three approaches in terms of fuel consumption the results were normalized using the commercial method as reference (100%). Results in Table 7.4, in which smaller values represent better solutions, show that the performance of the three systems is comparable for the small pipeline sub-network (SN1). Note that IOPO is substantially better for the larger and more complex sub-network which has higher throughput levels and higher fuel consumption than the smaller one. For the SN2/MF instance IOPO's fuel consumption is 12% better (lower) than the commercial method which would amount to an operating cost reduction of $28.7 Million per year for the combined cost of fuel gas and the environmental impact of CO_2 emissions at current market prices ($7/ GJ and $15/Tonnes of CO_2 respectively). If the prices increase the operating cost reduction increases. The fuel consumption differences between the optimization methods are for ideal pipeline operating conditions which almost never occur. However, the results can be used in the development of operating strategies which can take advantage of this value opportunity. Even if the transportation industry can capture 10% of the potential dollar saving improvements,

Table 7.3: Comparison with regard to Run time (wall clock time).

Instance	MOGA	Commercial Method	IOPO
SN1/HF	0:25:00	0:05:31	0:04:19
SN1/MF	0:26:15	0:05:38	0:02:32
SN1/LF	0:23:00	0:05:27	0:01:59
SN2/HF	107:46:00	0:43:09	0:29:02
SN2/MF	107:49:01	0:49:43	0:27:06
SN2/LF	99:14:00	0:49:09	0:28:19

the results are still substantial.

7.3 Evaluation of the Multi-Agent/Hardware Platform Aspect

As mentioned in Section 7.1.2, the available hardware for this thesis is a workstation with two dual cores. The next series of experiments aim at evaluating the multi-agent aspect of the IOPO system, more precisely the influence of the number of agents used on the results. We concentrate our analysis only on the search instances for SN2 since the runtimes for SN1 are too short to see any significant changes.

As stated in our introduction, a big draw towards using multi-agent search approaches for hard optimization problems in industry is the seemingly obvious fit for

Table 7.4: Comparison with regard to fuel consumption (Commercial Method = 100%).

Instance	MOGA	Commercial Method	IOPO
SN1/HF	101	100	99
SN1/MF	104	100	104
SN1/LF	102	100	102
SN2/HF	90	100	86
SN2/MF	96	100	88
SN2/LF	92	100	86

multi-processor multi-core workstations in order to use the number of available processors to speed-up the search. In order to analyze this aspect of our system, we ran experimental series for 1, 2, 3 and 4 search agents in IOPO (n_{MASS} = 1, 2, 3 and 4 respectively), where for each number of search agents the total number of solutions created –and therefore the number of calls to the hydraulic simulator– were the same, namely 4000, which was also the number used for the experiments in Section 7.2 for SN2. This means that with one search agent this search agent performs 400 updates of the 10 particles, with 2 search agents each search agent does 200 updates of their 10 particles and so on. For 2, 3, and 4 search agents we fit their searches into 10 rounds, so that for two search agents the communication between the search agents

takes place every 20 updates up to communicating after every 10 updates for 4 search agents. This way, the communication overhead is essentially the same for each number of search agents and we can get a good picture regarding the utilization of the processors by IOPO. As in the previous subsection, we report the average runtime over 10 runs.

Table 7.5 shows the results of the experimental series described above. As can be seen, the results are rather disappointing. While using 2 search agents gives us quite some speed-up despite the additional communication effort, adding the 3^{rd} and the 4^{th} search agent does not really accomplish a lot. The question from the multi-agent perspective is now, if this disappointing outcome is due to the multi-agent search approach, i.e. TECHS/IOPO, or due to the combination of this approach with a multi-processor multi-core hardware platform. While we did not use all of the features of TECHS as described in [Denzinger and Fuchs 1999] and [Denzinger and Offermann 1999] by not using heterogeneous search agents we definitely reduced the possibility for synergetic effects, and it is a well-known fact that for each cooperative search scheme and each problem instance there is a maximum number of processors beyond which no gains in speed can be achieved, we nevertheless were not convinced that this number is 2 for all the search instances we were looking at. Therefore we suspected the hardware platform to be the problem and in order to prove this we switched, as proof of concept, to a network of 4 old Pentium III with 256 MB of RAM each that we had available.

Table 7.6 presents the runtime comparison using the Pentium III computers. As

Table 7.5: Runtime comparison for different numbers of search agents (in minutes), using a 2 Dual Core workstation)

Instance	1 search agent	2 search agents	3 search agents	4 search agents
SN2/HF	54	33	31	30
SN2/MF	50	29	28	27
SN2/LF	56	33	31	29

can be seen, adding a 3^{rd} and 4^{th} search agent, i.e. processor, now has still quite an impact on the runtimes ruling out the TECHS/IOPO approach as sole problem. We can not expect linear improvements when adding more search agents/processors in our setting simply because with a smaller number of particle updates the mes_i-function of a search agent $\mathcal{A}g_i$ does have to choose from many results, thus providing less useful information to other search agents, respectively really useful information only later during a run. Therefore the results of Table 7.6 reflect well the behavior we expect from a homogeneous version of TECHS.

But why does changing from the multi-processor multi-core platform to a multi-computer platform produce a so different behavior? While we were able to assign the search agents to different cores, we do not know what the operating system does with the simulator processes. But even more important, all cores do share the same periphery, i.e. memory that is not on the processor chip and access to the file system. Given the fact that the pipeline simulations produce substantial file traffic and also require

Table 7.6: Runtime comparison for different number of search agents (in minutes), using 4 Pentium IIIs)

Instance	1 search agent	2 search agents	3 search agents	4 search agents
SN2/HF	240	125	92	69
SN2/MF	220	116	82	63
SN2/LF	225	132	83	65

significant memory, this might produce system level bottlenecks that result in the observed disappointing behavior. This requires additional research and at the moment we have to accept the fact that we might have to include systems level programming into the search agents to overcome this.

While the utilization of the available processors is not a strong point of IOPO at least for multi-processor multi-core hardware architecture, speed-up is not the only thing that is of interest for our industry partner. Most set-based search approaches make considerable use of random factors, so that two runs of a system using such approaches usually lead to two different solutions being found as the best solution of a run. How much runs vary with this regard is of quite some importance for a user. Therefore we compared the results from Table 7.5 with regard to how much the runs vary in their solution quality.

Figures 7.3, 7.4 and 7.5 provide the graphical representation of the variance over 10 runs in solution quality (normalized fuel consumption) in our experiments for the

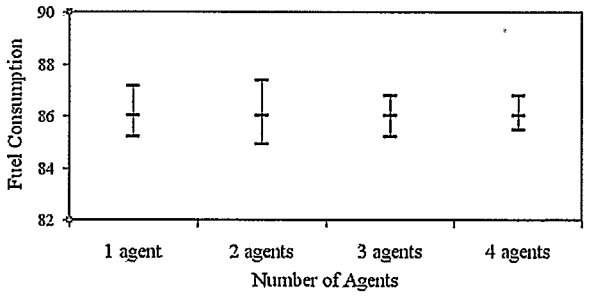


Figure 7.3: Variance in solution quality for different number of search agents (HF).

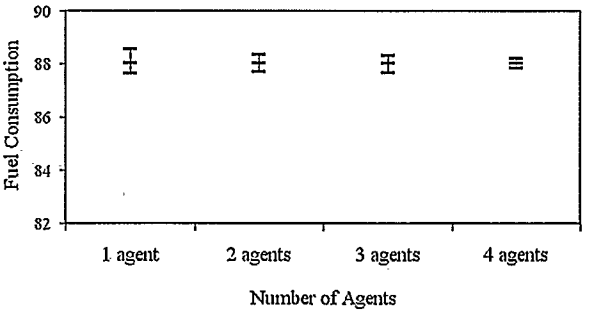


Figure 7.4: Variance in solution quality for different numbers of search agents (MF).

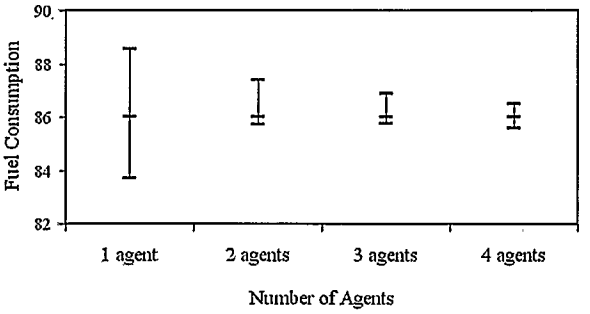


Figure 7.5: Variance in solution quality for different numbers of search agents (LF).

HF, MF, and LF instance respectively. As the figures show, using more search agents has a positive effect here since the variance is reduced with each search agent added to the system. This means that the quality of the solutions produced by IOPO is more predictable when more search agents are used which, from an economical and planning point of view, is a very important observation.

Although it is out of the scope of this thesis, additional testing could be done to evaluate IOPO's performance across a variety of serial, parallel, and distributed architectures. Architectural characteristics of interest may include network topology, processor speed, memory access, I/O, and the like.

7.4 IOPO for Multi-Objective Optimization of NGPO

In Section 7.2 we compared the performance of IOPO against two other approaches when applied to the fuel (single-objective) optimization problem of NGPO. In this section we present IOPOs performance for an even more challenging test: the multi-objective optimization of NGPO. The experiments in this section concentrate on the minimization of fuel consumption ($Min\ f_1(\vec{x})$) while maximizing the throughput ($Max\ f_2(\vec{x})$) for steady state simulations of natural gas pipeline networks.

At the time of this thesis there were no other methods for multi-objective optimization of NGPO available to compare against IOPO. The commercial method used for comparison in Section 7.2 is restricted to single objective optimization problems. MOGA on the other hand, has the capability to treat multi-objective problems but

proved to be extremely slow for the single-objective NGPO optimization problem so that it was not worthwhile to test it for the MOO case.

The MOO experiments in this section are aimed to evaluate the performance of the two-phase approach introduced in Chapter 5 to optimize the operations of the largest and more complex pipeline sub-network, SN2. Note that, in contrast to the 3 flow instances (HF, MF, LF) used in the experiments of Section 7.2, the 2-objective scenario in this section considers the complete spectrum of flow levels, from the lowest flow level to the highest flow level. With the proposed two-phase approach we expect the $\mathcal{MASS}$ to be able to find solutions (Pareto front) to the given instance of the search problem not only faster but with better quality than if using only PHASE II (referred as PHASE II-Only).

In order to make a fair comparison between both methods we should let both either: 1) use the same number of transitions to solve the problem, or 2) let them run for the same amount of time and then compare their solutions. We selected the first option and set an approximately equal maximum number of iterations for each of the methods, see Table 7.7. The IOPO parameters for PHASE II-Only are $n_{\mathcal{MASS}}$=4, n_{swarm}=36, n_{init}=62, $n_{sol_i^+}$=[20,30,40,50], $n_{k_i^+}$=[70,50,30,20,10], $\mathcal{G}$=207. The IOPO parameters for the two-phase IOPO approach for MOO are: for PHASE I $n_{\mathcal{MASS}}$=4, n_{swarm}=10, n_{init}=15, $n_{sol_i^+}$=[10,15,20], $n_{k_i^+}$=[30,15,5], $\mathcal{G}$=50, n_{Pr}=12, and for PHASE II are $n_{\mathcal{MASS}}$=4, n_{swarm}=20, n_{init}=21, $n_{sol_i^+}$=[10,15,20,25], $n_{k_i^+}$=[15,20,15,10,5], $\mathcal{G}$=72. Figure 7.6 shows the results, the non-dominated fronts, found by the two approaches:

Table 7.7: Results of Multi-Objective Optimization of SN2

System	Max. Number of Transitions	Run Time (min)
IOPO (Two-Phase Approach)	29,760	228
IOPO (PHASE II-Only)	29,808	200

- PHASE II-Only (triangles)
- Two-phase IOPO approach for MOO (filled circles)

where the x-axis indicates the total amount of fuel consumed $f_1(\vec{x})$ by the compressor units that are being used in all those compressor stations that are ONLINE. The y-axis is the throughput $f_2(\vec{x})$ at a node of interest.

There is an obvious difference in the capability of the approaches to cover the complete front with a well distributed set of solutions. Note that the solutions found by PHASE II-Only are concentrated in the lower-left area. There are three important observations about these results. First, the pipeline system has considerable more flexibility, e.g. more valid configurations, to operate when the throughput level is low; this is the reason why it is easier to find solutions in this area of the search space and as a consequence this part of the Pareto front is quickly populated[21]. The second observation is related to the discontinuities observed in the Pareto front obtained by

[21]It is also important to mention that the hydraulic simulation runs consume more time for higher throughput levels than for lower throughput levels.

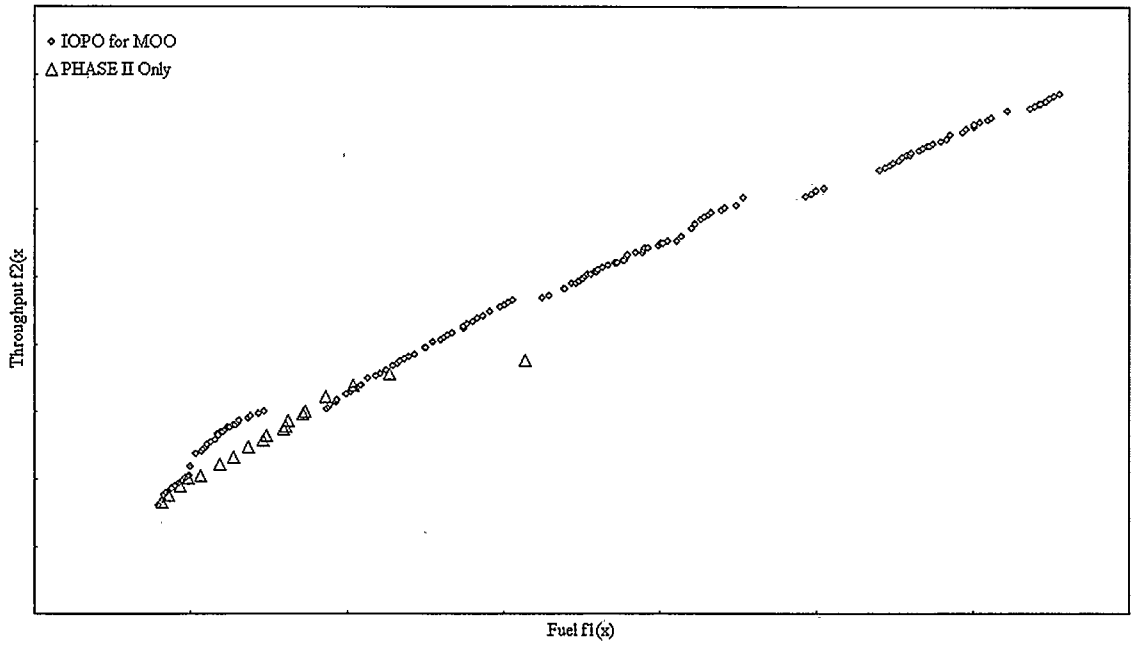


Figure 7.6: Multi-Objective Optimization of SN2 = $Min\ f_1(\vec{x})$, $Max\ f_2(\vec{x})$

the two-phase approach. Careful examination of the sets of operating points at the edging points of the gaps reveals simultaneous changes in the status of one or even two compressor stations from one solution to the nearest one, just right after the gap. The third observation is about the number of solutions allowed in the Pareto front. As introduced in Section 2.1, this number can be controlled by controlling the maximum capacity of the archive that stores the set of non-dominated solutions. In this case, for the two-phase approach increasing the size of the archive would not provide additional information about the system, just redundant solutions.

Now, to put the reader in perspective, we mention the differences between our two-phase IOPO approach and a previous attempt to solve the MOO for NGPO problem

using MOGA. In the work reported in [Botros et al. 2004], MOGA was applied to solve the MOO NGPO problem (*Min* fuel, *Max* throughput) for an instance of similar complexity to our SN2 (53 decision variables). The results of the experiments demonstrated difficulty with the genetic algorithms parameter selection as different parameter selection provided different solutions, i.e. different Pareto fronts. The experiment that provided the best (but still incomplete) Pareto front required 100,000 evaluations using a population size of 500 and a total of 200 generations[22]. Unfortunately no run time was reported in the results, but results of our experiments demonstrate that the two-phase IOPO approach can find the complete Pareto front using less than 30,000 evaluations (Table 7.7), which directly translates into a reduced runtime compared to MOGA[23].

Later on, in [Botros et al. 2006] the authors provided an improved solution approach based on the use of surrogate methods. The results of the experiments for two-objective optimization only report MOGA's performance for a small instance of similar complexity to SN1 (13 decision variables) showing again difficulty to approximate the complete Pareto front. The best run time reported on this significantly smaller pipeline subnetwork was 3.05 hours (183 minutes)[24] compared to the 228 minutes IOPO needs to solve the larger and more complex SN2 pipeline subnetwork (53 decision variables). Unfortunately no computational time was reported in the publication for the larger

[22]Pareto front data shown in Figure 9 in [Botros et al. 2004].

[23]A reminder that the function evaluations are the most time consuming part of the search, and that the same hydraulic simulator was used in the MOGA and IOPO experiments.

[24]Computational time reported in Table 6 in [Botros et al. 2006].

network similar to our SN2.

The results of the experiments in this section demonstrate that, although the form that each search agent in the $\mathcal{MASS}$ interacts with the hydraulic simulator is not the most desirable[25], the algorithmic design aspects of IOPO for MOO decreases the computational cost and makes it efficient to solve the complex problem of MOO of NGPO and the solution is obtained in an acceptable time frame.

[25]Reading and writing files is the only way to communicate with the hydraulic simulator used in this thesis.

8 Conclusions and Future Work

This chapter summarizes the proposed approach to optimize natural gas pipeline operations and highlights the results achieved from the experiments in terms of execution time and quality of the solution. This chapter also includes potential directions for future work that were identified during the course of this research.

8.1 Summary

The main objective of this thesis is to reduce the computational cost required to optimize the operation of large and complex natural gas pipeline networks while maintaining not only the targeted throughputs and delivery contracts but also meeting the standards and regulations aimed to reduce the inherent risk of high-pressure transportation systems.

To achieve this goal we proposed a biologically-inspired computational model in the form of a multi-agent cooperative search system. The search agents in the system are designed to perform the search and communicate periodically to exchange solutions, some of which are incorporated in the search agents' search state and others are used

to improve their search control. The approach exploits the strength of particle swarm optimization to deal with high-dimensional problems that include a mix of discrete and continuous decision variables. Domain knowledge was incorporated into the search to improve its efficiency. This permitted the identification of solutions that can be evaluated without having to use the time-intensive pipeline simulations used to measure the quality of a solution and the creation of a lexicographical combination of orderings to compare solution vectors to guide the search more efficiently. The use of the multi-agent approach showed to be advantageous with regard to reducing the solution quality variance between several runs, something crucial for the industrial users of the search system. We also extended the proposed approach to the multi-objective optimization case.

The experimental evaluation was done using six problem instances from two pipeline sub-networks of TransCanada PipeLines Ltd.. Results showed that the IOPO system is able to optimize real world pipeline sub-networks for various flow requirements within the 30 minutes time limit that is considered reasonable. The improved performance of IOPO is noticeable in the problem instances that involve the large and complex pipeline sub-network in which it reliably outperforms the interactive method currently considered as state-of-the-art in the transportation industry by producing solutions that are at least 12% better [Mora et al. 2008a, Mora et al. 2008b]. Given the considerable large volume of natural gas that is transported in modern transmission lines, the improved solution obtained with IOPO can be translated into a considerable reduc-

tion in operation and maintenance costs and a significant reduction of greenhouse gas emissions. It is important to mention that, in contrast to the requirements imposed by the commercial system, IOPO achieves these results without any human interaction making the quality of the solution independent of the experience of the user.

8.2 Future Work

The research reported in this thesis is only the first step in exploiting the potential that multi-agent cooperative search with biologically-inspired methods has for solving hard optimization problems in industry. Our PSO approach allows for much more customization to the needs of TransCanada PipeLines Ltd. than what we have realized so far. Good solutions from system runs for similar problems can be used to guide the swarm faster towards good solutions. The currently homogeneous processes used to make use of the multi-processors and multi-cores can be enhanced by allowing the use of heterogeneous search processes with other biologically-inspired techniques such as genetic algorithms.

With further speed improvements comes the possibility to try out different boundary conditions for pipeline sub-networks which should be undertaken by having the distributed solution processes for each pipeline sub-network cooperate on an additional level, thus providing the human decision makers with better support for their negotiations regarding these boundary conditions. Biologically-inspired methods also offer the possibility of multi-objective optimization, which is a second direction leading

nearer towards the real decision making process that TransCanada PipeLines Ltd. is employing. The proposed approach can be used to find either the minimum or the maximum linepack levels for different flow conditions and provide what is called the *linepack envelope* which indicates the valid operating region from which the best operational settings (those that lead to the least amount of fuel consumed by compressor stations) can be identified.

This research will be continued to use IOPO for different connected pipeline subnetworks of the TransCanada PipeLines Ltd. system. The results would reinforce the viability and robustness of IOPO and the potential for deployment of the system.

We are pleased to know that the industry is willing to assess non conventional approaches such as our IOPO system to tackle problems like the optimization of pipeline operations. We hope that the application of this research encourages the trial of these approaches for other real world problems as these challenging problem push the development of new methods. The proposed approach in this thesis can be used as the basis to treat problems with similar characteristics. Just for the case of pipeline operations, the application can be extended to the transportation of other fluids such as oil and its derivates by making the necessary adaptations, e.g. by using equations for incompressible fluids, using pumping stations instead of compressor stations, etc.. By the same token, this method can be a reasonable solution for similar problem instances in water distribution networks.

Bibliography

Bolkan, Y. G. (1991). An efficient algorithm for optimal pipeline design and operation. Master's thesis, University of Calgary.

Botros, K. K., Sennhauser, D., Jungowski, K., Poissant, G., Golshan, H., and Stoffregen, J. (2004). Multi-objective optimization of large pipeline networks using genetic algorithms. In *International Pipeline Conference 2004*, Calgary, Canada. ASME.

Botros, K. K., Sennhauser, D., Stoffregen, J., Jungowski, K., and Golshan, H. (2006). Large pipeline network optimization - summary and conclusions of transcanada research effort. In *International Pipeline Conference 2006*, Calgary, Canada. ASME.

Coello, C. A. and Salazar-Lechuga, M. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *In Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1051–1056, Piscataway, New Jersey. IEEE Service Center.

Coello, C. A., Toscano-Pulido, G., and Salazar-Lechuga, M. (2002). An extension of particle swarm optimization that can handle multiple objectives. In *Workshop on*

Multiple Objective Metaheuristics, Paris, France.

Coello, C. A., Toscano-Pulido, G., and Salazar-Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions On Evolutionary Computation*, 8(3):256–279.

Coello, C. A. C. (2006). Evolutionary multi-objective optimization: A historical view of the field. In *IEEE Computational Intelligence Magazine*, volume 1, No. 1, pages 28–36.

Dandy, G. C., Simpson, A. R., and Murphy, L. J. (1996). An improved genetic algorithm for pipe network optimization. *Water Resources Research*, 32(2):449–458.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons.

den Berg, F. V. (2002). *An Analysis of Particle Swarm Optimizers.* PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

Denzinger, J. (1999). *Distributed Knowledge-based Search.* PhD thesis, University of Kaiserslautern, Kaiserslautern, Germany.

Denzinger, J. (2000). *Conflicting Agents: Conflict management in multi-agent systems*, chapter Conflict Handling in Collaborative Search, pages 251–278. Kluwer Academic Publishers.

Denzinger, J. and Fuchs, D. (1999). Cooperation of heterogeneous provers. In *International Joint Conference on Artificial Intelligence IJCAI-99*, pages 10–15.

Denzinger, J. and Offermann, T. (1999). On cooperation between evolutionary algorithms and other search paradigms. In *Congress on Evolutionary Computation 1999*, pages 2317–2324. IEEE-Press.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micromachine and Human Science*, pages 39–43.

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons.

Eusuff, M. M. (2004). *Water Resources Decision Making using Meta-Heuristic Optimization Methods.* PhD thesis, Department of Civil Engineering and Engineering Mechanics, University of Arizona.

Foundation, P. C. (1999). *Our Petroleum Challenge: Exploring Canadas Oil and Gas Industry.* Petroleum Communication Foundation.

Goldberg, D. E. (1983). *Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning.* PhD thesis, The University of Michigan.

Goldberg, D. E. (1987a). Computer-aided pipeline operation using genetic algorithms and rule learning. part i: Genetic algorithms in pipeline optimization. *Engineering with Computers*, 3:35–45.

Goldberg, D. E. (1987b). Computer-aided pipeline operation using genetic algorithms and rule learning. part ii: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers*, 3:47–58.

Goslinga, J., Kaulback, M., Witczak, K., and McNeill, B. (1994). A method for pipeline network optimization. In *13th OMAE Offshore Mechanics and Arctic Engineering Conference*, volume 5, pages 31–43. Pipeline Technology, ASME.

Ho, S. L., Shiyou, Y., Guangzheng, N., Lo, E. W. C., and Wong, H. C. (2005). A particle swarm optimization-based method for multi-objective design optimization. *IEEE Transactions on Magnetics*, 41(5):1756–1759.

Hu, X. and Eberhart, R. (2002). Multi-objective optimization using dynamic neighborhood particle swarm optimizaiton. In et. al., D. B. F., editor, *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1677–1681. IEEE.

Janson, S. and Merkle, D. (2005). A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking. In Blesa, M. J., Blum, C., Roli, A., and Sampels, M., editors, *Hybrid Metaheuristics, Second International Workshop*, volume 3636 of *Lecture Notes in Computer Science*, pages 128–142. Springer.

Jung, B. S. and Karney, B. W. (2004). Transient state control in pipelines using gas and particle swarm optimization. In Liong, Phoon, and Babovic, editors, *6th International Conference on Hydroinformatics*. World Scientific Publishing Company.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN'95)*, volume 4, pages 1942–1948. IEEE Service Center.

Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, pages 4104–4109.

Kerkhof, J., Williams, L., Mah, K., and Kmet, J. (2002). Transcanada pipelines: An expert advisory system for pipeline operations. In *International Pipeline Conference*. ASME.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Krishnaswami, P., Chapman, K. S., and Abbaspour, M. (2004). Fuel-efficient operation of compressor stations using simulation-based optimization. In *International Pipeline Conference*, number IPC04-0113. ASME.

Liu, J., Jing, H., and Tang, Y. (2002). Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136(1):101–144.

Maier, H. R., Simpson, A. R., Zecchin, A. C., Foong, W. K., Phang, K. Y., Seah, H. Y., and Tan, C. L. (2003). Ant colony for design of water distribution systems. *Journal of Water Resources Planning and Management ASCE*, pages 200–209.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer-Verlag, Berlin, 3rd revised and extended edition.

Mohitpour, M., Golshan, H., and Murray, A. (2003). *Pipeline Design & Construction: A Practical Approach.* ASME.

Mora, T., Denzinger, J., and Golshan, H. (2008a). Cooperative search for optimizing pipeline operations. In Berger, Burg, N., editor, *7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Industry and Applications Track, Estoril, Portugal. International Foundation for Autonomous Agents and Multiagent Systems.

Mora, T., Sesay, A., Denzinger, J., Golshan, H., Poissant, G., and Konecnik, C. (2008b). Fuel optimization using biologically-inspired computational models. In *International Pipeline Conference 2008*, Operations & Maintenance, Calgary, Canada.

Mostaghim, S. and Teich, J. (2004). Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *Congress on Evolutionary Computation (CEC 2004)*, volume 2, pages 1404–1411.

of America (IPAA) Americas Oil, I. P. A. and Producers, G. (2004). Economic reports,. Website. `http://www.ipaa.org/info/econreports/usps.asp?Table=Chart12`.

on Trade, U. N. C. and UNCTAD, D. (2004). United nations conference on trade and development unctad. Website. http://r0.unctad.org/infocomm/anglais/gas/market.htm.

Ozcan, E. and Mohan, C. K. (1998). Analysis of a simple particle swarm optimization system. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 253–258.

Ozcan, E. and Mohan, C. K. (1999). Particle swarm optimization: Surfing the waves. In *Congress on Evolutionary Computation*, pages 1939–1944. IEEE Press.

Percell, P. B. and Ryan, M. J. (1987). Steady-state optimization of gas pipeline network operation. In *Pipeline Simulation Interest Group Annual Meeting*, number 8703, Tulsa, USA.

Ratnaweera, A. C., Halgamuge, S. K., and Watson, H. C. (2002). Particle swarm optimizer with time varying acceleration coefficients. In *International Conference on Soft Computing and Intelligent Systems*, pages 240–255.

Reyes-Sierra, M. and Coello, C. A. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *Journal of Computational Intelligence Research*, 2(3):287–308.

Rios-Mercado, R. Z. (2002). *Handbook of Applied Optimization*, chapter Natural Gas Pipeline Optimization. Oxford University Press.

Rios-Mercado, R. Z., Wu, S., Scott, L. R., and Boyd, E. A. (2002). A reduction technique for natural gas transmission network optimization problems. *Annals of Operations Research*, 117(1).

Shi, Y. and Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pages 591–600.

Simpson, A. R., Dandy, G. C., and Murphy, L. J. (1994). Genetic algorithms compared to other techniques for pipe optimization. *Journal of Water Resources Planning and Management ASCE*, 120(4):423–443.

Stoffregen, J., Botros, K., Sennhauser, D., Jungowski, K., and Golshan, H. (2005). Pipeline network optimization application of genetic algorithm methodologies. In *Pipeline Simulation Interest Group*, number 0502, San Antonio, USA.

Suganthan, P. N. (1999). Particle swarm optimiser with neighborhood operator. In *IEEE Congress on Evolutionary Computation*, pages 1958–1962.

Suttner, C. B. (1995). *Parallelization of Search-based Systems by Static Partitioning with Slackness.* PhD thesis, Institut fur Informatik, Technische Universitat Munchen. Published by Infix-Verlag, Volume DISKI 101.

Toscano-Pulido, G. and Coello, C. A. (2004). Using clustering techniques to improve the performance of a particle swarm optimizer. In *Genetic and Evolutionary Com-*

putation Conference GECCO, volume 3102 of *Lecture Notes In Computer Science*, pages 225–237.

VanVeldhuizen, D., Zydallis, J., and Lamont, G. (2003). Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):144–173.

Veloso, B. C., Pires, L. F. G., and Azevedo, L. F. A. (2004). Optimization of pump energy consumption in oil pipelines. In *International Pipeline Conference*, Calgary, Canada. ASME.

Wong, P. J. and Larson, R. R. (1968). Optimization of natural-gas pipeline systems via dynamic programming. *IEEE Transactions on Automatic Control*, 13(5):475–481.

Wright, S., Somani, M., and Ditzel, C. (1998). Compressor station optimization. In *Pipeline Simulation Interest Group*, number 9805, Denver, USA.

Wu, S. (1998). *Steady-State Simulation and Fuel Cost Minimization of Gas Pipeline Networks*. PhD thesis, Department of Mathematics, University of Houston.

Xiao-hua, Z., Hong-yun, M., and Li-cheng, J. (2005). Intelligent particle swarm optimization in multiobjective optimization. In *Congress on Evolutionary Computation (CEC 2005)*.

Zhang, Z. (1999). Fluid transients and pipeline optimization using genetic algorithms. Master's thesis, Graduate Department of Civil and Environmental Engineering, University of Toronto.

Zitzler, E. and Thiele, L. (1999). Multi-objective evolutionary algorithm: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3 (4)(4):257–271.

Zitzler, E., Thiele, L., and Deb, K. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.