OPTIMIZATION OF IDEAL REACTIVE DISTILLATION COLUMN USING HYBRID PSO ALGORITHM

BY

MOHAMMED MIRZA AMER BAIG

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

JANUARY, 2006

DEDICATED TO

LATE FAJJI & LATE AMMI

ACKNOWLEDGMENT

All praise is to Allah (Subhanahu Wa Ta'aala) with whose mercy and blessings I have achieved what I have achieved today. Peace and blessings of Allah be upon the final prophet.

I take this opportunity to thank my advisor Dr. Hussain Nasir Al-Duwaish. for his guidance and motivation which he has extended to me throughout my thesis and during my stay at this university. I am grateful to him for his unlimited favors and help which he extended to me during my rough time in my life and for his help in the administrative affairs of mine related to the university. I extend my gratitude to my Co-Advisor, Dr. Muhammad A. Al-Arfaj, for his help and guidance in making my dream come true. I am grateful to him for his time which he has given to me without bounds for the completion of my thesis and for his faith and trust which he had in me during my work with him.

I gratefully acknowledge Dr. Al-Baiyat Samir A. for his advice and guidance through my research and Dr. Bakhashwain Jamil M. for his profound support and Dr Mantawy Abdel-Aal H. for his help in my work.

I further extend my thanks to Moshood who has been more than a help ever since I started my work in this area and has always been there to help me anytime I need him. I

thank my department and especially my chairman for providing me with abundant facilities of PC-Labs for simulation and helping me in my issues.

It would be ungrateful if I forget my family at this time, I am grateful to my father, mother and my sisters for their constant prayers their patience and faith in me, a pillar of support ever since I was born. I also extend my thanks to Liaqat bhaiya, Mahamid Bhaiya, Ismail Bhai, Ashraf Bhaiya.

Last but not the least my armada of friends who always kept me on the edge of my seat and provided me with every thing I needed to realize this dream. To mention a few: AmerBhai, Faisal, Ayub, Abbas, Baber, Riyaz, Abdul Hameed, FasiBhai, AneesBhai, Abdul Hafeez(Shaukat), Baba, Aleem, Ilyas, Wajid, Mansoor, Kareem, Rahman, Asif, Asim, Kazim, Rahim, Tanveer, Mubashir, Guddu, Aslam, Faizan, Allauddin. And seniors SohelBhai, RazaBhai, BasharBhai, TayyabBhai and Nasir Bhai(AmeerSaab).

Special thanks to Sheikh Saleh Al-Munajjid, Moulana Abdul Quavi Sahab, Dr. Hafiz Afzal Sahab, Dr. Zafarullah Khan Sahab, Dr. Abdul Aziz Sahab and My masters who taught me to read Holy Qur'aan.

Thank you one and all.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: **MOHAMMED MIRZA AMER BAIG**

TITLE: **OPTIMIZATION OF IDEAL REACIVE DISTILLATION COLUMN USING HYBRID PSO ALGORITHM.**

DEPARTMENT: **ELECTRICAL ENGINEERING**

DATE: **JANUARY, 2006**

Optimization has developed into a prime field of engineering. Its applications can be widely seen in industrial as well as research sectors. Particle Swarm Optimization (PSO) is one of the various tools used in research and industry to maximize efficiency out of process or industry by optimization. PSO has gained much credit due to its simplicity and faster convergence compared to other evolutionary algorithms. However there are certain limitations that need to be overcome like premature convergence and decreased acceleration when reaching the optimal solution. In this thesis, Hybrid-Integer Particle Swarm Optimization Algorithm is developed which can optimize mixed Integer problems with a very fast pace and sure convergence.

An example of an industrial process is Reactive Distillation Column (RDC). It has modernized the petrochemical industry in the area of separation of products by the methods of distillation. Distillation requires higher temperature to be maintained in RDC, the burners that supply the necessary heat to maintain the temperature require constant supply of fuel. RDC exhibits non-linearity due to chemical reactions. The design of the RDC is a factor that determines the amount of fuel consumed. Thus optimization at design level will lead to optimal fuel consumption of process. The developed Hybrid-Integer PSO in this research is applied to find an optimal design for minimum fuel consumption and attain the same purity of separated products.

It is noticed that the optimization leads to a process design with lesser fuel consumption and same product purity. The optimization algorithm is also tested for standard test problems and the results are compared to those cited.

MASTER OF SCIENCE DEGREE
KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA

THESIS ABSTRACT (ARABIC)

ملخص الرسالة

الاسم :محمد ميرزا أمير باج
عنوان الرسالة :التصميم الأمثل لعملية صناعية باستخدام **Hybrid PSO Algorithms**
التخصص :الهندسة الكهربية
تاريخ التخرج : يناير 2006

أصبح الوصول للتصميم الأمثل أحد المجالات الرئيسية في الهندسة التي لها تطبيقات واسعة في الصناعة والنطاق البحثي. أحد الأدوات المستخدمة في المجال الصناعي والبحثي للحصول على اكبر كفاءة ممكنة لعملية أو صناعة ما هو Particle Swam Optimization (PSO) . اكتسبت هذه الطريقة أهمية كبيرة بسبب بساطتها وسرعة الوصول للنتائج مقارنة بالطرق القديمة لكن توجد لهذه الطريقة بعض القيود التي يجب أن تعالج. في هذه الرسالة تم تطوير Hybrid-Integer Particle Swarm Optimization Algorithm لمعالجة المشاكل الصحيحة المختلطة بخطوة سريعة جدا وتقارب أكيد.

Reactive Distillation Column (RDC) هو أحد الأمثلة على العمليات الصناعية التي طورت الصناعات البتروكيماوية في مجال فصل المنتجات بطريقة التقطير. يتطلب التقطير توفير درجة حرارة عالية والتي بدورها تتطلب كمية وقود مناسبة. عملية ال (RDC) تتميز بأنها معقدة وغير خطية بدرجة كبيرة وباعتماد كمية الوقود المستهلكة على التصميم . الوصول للحل الأمثل في مرحلة التصميم بالتالي يؤدي إلى الاستهلاك المثالي للوقود. في هذا البحث تم استخدام طريقة ال Hybrid-Inteeger PSO للوصول للتصميم الأمثل لاستهلاك أقل كمية من الوقود والحفاظ على نفس درجة نقاوة المنتجات التي يتم فصلها. تم اختبار هذه الطريقة أيضا في الوصول للتصميم الأمثل لبعض المشاكل القياسية والنتائج كانت مقاربة للنتائج المنشورة.

درجة ماجستير
جامعة الملك فهد البترول و المعدن
الظهران - 31261
المملكة العربية السعودية

CHAPTER 1

INTRODUCTION

1.1 Introduction and Motivation

Optimization is one of the major quantitative tools in the machinery of decision-making. A wide variety of problems in the design, construction, operation and analysis of industrial process can be resolved by optimization [**1**]. A typical engineering problem can be posed as follows: given a process that can be represented by mathematical equations and a performance criterion such as minimizing annual cost of maintenance. The goal of optimization is to find the values of the variables in the process that yield the best value of the performance criterion. Together the process and the performance criterion represent the optimization problem [**1**]. One such tool of optimization which has gained high recognition recently is Particle Swarm Optimization which is generally referred to as PSO.

The industrial sector has always sought after reducing the capital investment and maximizing the profits based on the investments. An important and ever growing sector in the industry is the "Reactive Distillation" in the chemical industry. Reactive Distillation is used to separate chemical reactants after the reaction into different

products. Its importance and benefits are being realized everywhere in the world due to its single column reaction and separation ability. This compact nature of Reactive Distillation has led to its important role and has obliged the industry to switch from classical methods of separate reaction and distillation to a compact one "Reactive Distillation Column" where both the process of reaction and separation take place [2]. A commending effort has been done by Al-Arfaj and Luyben in designing an ideal hypothetical reactive distillation column. Where all the complexities are stripped aside and component control of the process is given importance [20]. One of the major requirements of the "Reactive Distillation Column" is that it requires a constant supply of heat so as to maintain the temperature of reaction in the entire section. This requires a heat source which generally is a burner and is called reboiler.

Little has been done in the research area to develop a model which can provide an optimal design which consumes least amount of fuel required by the burner. With the rise in the cost of oil and its ever increasing demand, it is of great significance to have a model that consumes least amount of fuel but delivers the exact concentration of products.

This research is a step towards the development of optimal reactive distillation column so as to reduce the capital incurred on the fuel consumption. A floating model is step wise developed out of an existing model [20] thereafter a hybridized version of PSO is applied

to find the optimal design parameters like Trays, Pressure and the Feed locations in the reactive distillation column so as to produce the same products concentration with minimal of fuel consumption.

1.2 Literature Review

1.2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic optimization algorithm that belongs to the category of *swarm intelligence* methods [11]. PSO has attained increasing popularity due to its ability to solve efficiently and effectively a plethora of problems in diverse scientific fields [21]. Most of these problems involve the minimization of a static objective function, i.e., the main goal is the computation of a global minimizer that does not change.

The research has grown extensively in the field of optimization and particularly in the field of PSO due to its ever increasing popularity. All this started when the concept of function-optimization by means of a particle swarm was introduced by James Kennedy, Russel and Eberhart [12]. Later they developed the algorithm and laid principles to what is today called as *Swarm Intelligence* [11], In which they defined the five basic principles on which the Swarm Intelligence was introduced.

Angeline [13] and Kennedy [14] discussed the drawback of simple PSO as was formulated first in comparison with the Evolutionary Algorithms. And it was not until the V_{max} operator was introduced by Kennedy [11] that made PSO worth to be recognized as a competitor to the evolutionary algorithms and Angeline [13] verified this with his comparisons. Later versions of Eberhart and Shi [18] introduced the concept of inertia weights where the velocities are multiplied by the inertia factor before updating and Maurice introduced constriction factor [15] which constricts the velocity to their present velocity by multiplying with the constriction factor before updating.

PSO converges very rapidly for uni-modal problems, but for multi-modal problems there is a great risk of the algorithm getting struck in the local optima, i.e. premature convergence, in order to avoid this one would have to look into all possible local optima before deciding on the global optimal value. This by itself is cumbersome the algorithm will take a large amount of time to cruise through all the local solutions before converging on the global optima. One such method to avoid this premature convergence was addressed by Lovberg [16] wherein they renewed the swarm by breeding some of the particles and called this algorithm the Hybrid-PSO a merger of Evolutionary algorithms with the basic PSO.

In order to improve the convergence speed of the PSO for multi-modal problems, Kennedy [11] proposed that if the an approximate clusters of particles which are assumed

to be near the global optima should replace the present global and local trajectories so that they can converge faster to the global optima.

Kennedy in [14] came up with the idea of neighborhood where in information between the particles could be shared so as to enhance the search space and achieve better convergence, e.g. In here the particles share the information with the two adjacent particles. He further discussed types of neighborhood techniques in his work which can influence the convergence.

The important criteria once the PSO algorithm is ready is to have a proper tuning of the parameters so that the global optimal solution can be achieved very fast and with out getting into the premature convergence pit. Several papers have mentioned the parameter selection criteria prominent among them are Shi and Eberhart[18][19] , Carlisle and Dozier [17], Clerc and Kennedy [15]and Angeline [13] a few out of vast researchers.

A wide variety of problems can be represented as discrete optimization models. Integer programming has many applications one among them is the training of neural networks with integer weights, where the activation function and weight values are confined in a narrow band of integers. Laskari et al. [36] developed Particle Swarm Optimization to handle integer problems and have compared their work with the classical Branch and Bound technique and concluded that PSO out performs Branch and Bound.

1.2.2 Reactive Distillation Column

Reactive distillation is the coupling of both physical separation and chemical reaction in one unit operation. It has been employed in industry for many decades, and its area of application has grown significantly. A reactive distillation column is usually split into three sections: reactive section, stripping section and rectifying section. In the reactive section, the reactants are converted into products, and where, by means of distillation, the products are separated out of reactive zone. The tasks of the rectifying and stripping sections depend on the boiling points of the reactant and product.

Several researchers have worked extensively on the conceptual design and process optimization of reactive distillation [1, 2]. Al-Arfaj and Luyben [3] have discussed the Effect of number of trays of fractionation on the performance of Reactive Distillation column and concluded that increasing or decreasing the fractionation trays does not degrade the performance maintaining the same concentration of the products.

Al-Arfaj and Luyben [20] studied the control of reactive distillation column that produced two products from a single reactive column by feeding exactly stoichiometric amount of the two fresh feed streams. They explored control structures, all of which included the measurement of composition of one of the reactants inside the reactive

section of the column. This composition is then used to adjust the appropriate fresh feed stream.

Olanrewaju [41] in his thesis developed linear online estimators to facilitate the measurement of samples under erroneous conditions and later developed estimator based control on reactive distillation column.

However, only a few papers have appeared that discuss the closed-loop optimization of reactive distillation column. Sane et al. [38] have shown that the introduction of a different tray holdup in the stripper and rectifier section of a continuous kinetically controlled reactive distillation column facilitates the design procedure. Their design allows minimizing investment related costs such as the column height and the amount of catalyst.

Cardoso et al. proposed an optimization model for the MINLP(Mixed integer Non-linear Programming) formulation of the reactive distillation columns using simulated annealing which gives optimal number of trays, optimal number of feed tray locations and composition profiles. Although the work is very interesting and looks concurrent to what we are about to propose it falls short of designing based upon the minimization of fuel consumption. Instead the material energy balance is used as a criterion to develop the design of reactive distillation model so as to obtain the desired concentration.

1.3 Objectives

Optimization has developed into a strong field of engineering. Its applications can be widely seen in Industrial as well as research sector. PSO is one of the many tools the research and industry are using so as to find out how best they can maximize the efficiency of their process or industry by conducting simulations in very short time. PSO has gained much of its credit due to its simplicity and faster convergence criteria compared to other evolutionary Algorithms. An example of a complex Non-Linear problem is Reactive distillation Column (RDC) where the hardware design of the process is a factor in the daily expenditure incurred.

Designing a Reactive Distillation Column by Mathematical tools is an option too. But the amount of time and working hours it takes has really turned the researchers to use optional methods, like the optimization methods, that have nothing to do with the actual design procedure but only need a cost function to minimize.

The present work develops a Hybrid Integer PSO and uses it for the application of such a problem where the design (of RDC) mainly affects the post installation expenditure. The number of trays which constitute the height, the operating pressure and the location of input feed are among the factors that affect amount of fuel intake which is a factor of

vapor boilup, necessary to maintain the temperature so as to perform the required reactions in the column. The specific objectives of this research in broad sense are

First: To find

- The Optimal Number of Stages in the Stripping(Bottom) Section
- The Optimal Number of Trays in the Reactive(Middle) Section
- The Optimal Number of Stages in the Rectifying(Bottom) Section

Such that the Cost of Energy (Vapor Boilup) is minimized and Concentrations of Products is 95%

Second: To find

- The Optimal Number of Stages in the Stripping(Bottom) Section
- The Optimal Number of Trays in the Reactive(Middle) Section
- The Optimal Number of Stages in the Rectifying(Bottom) Section
- Optimal Pressure in the Reaction Column

Such that the Cost of Energy (Vapor Boilup) is minimized and Concentrations of Products is 95%

Third: To find

- The Optimal Number of Stages in the Stripping(Bottom) Section
- The Optimal Number of Trays in the Reactive(Middle) Section
- The Optimal Number of Stages in the Rectifying(Bottom) Section
- The Optimal Pressure in the Reaction Column
- The Optimal Feed Location of First Feed
- The Optimal Feed Location of Second Feed

Such that the Cost of Energy (Vapor Boilup) is minimized and Concentrations of Products is 95%

1.4 Research Methodology

Briefly the research methodology can be illustrated as follows:

The Literature was reviewed in search of solution to integer problems then an Integer PSO model was developed. In order to avoid stagnation and enhance convergence towards the end of simulations Hybrid version of PSO was selected. The two models were used for developing a Hybrid-Integer PSO model with some changes. The RDC Model obtained from the literature is fixed. Any change in optimization Variables would cause change in the order of the system and the Non-Linear equations describing it. In order to use such a model an initial estimate of the composition profile is necessary. We

have developed a method which extrapolates the initial values based on the RDC system taken from the Literature. This estimate serves as a disturbance in the beginning to the system; the internal composition controllers control this error and bring the system back to steady state. The steady state operation of RDC is constant production at desired purity.

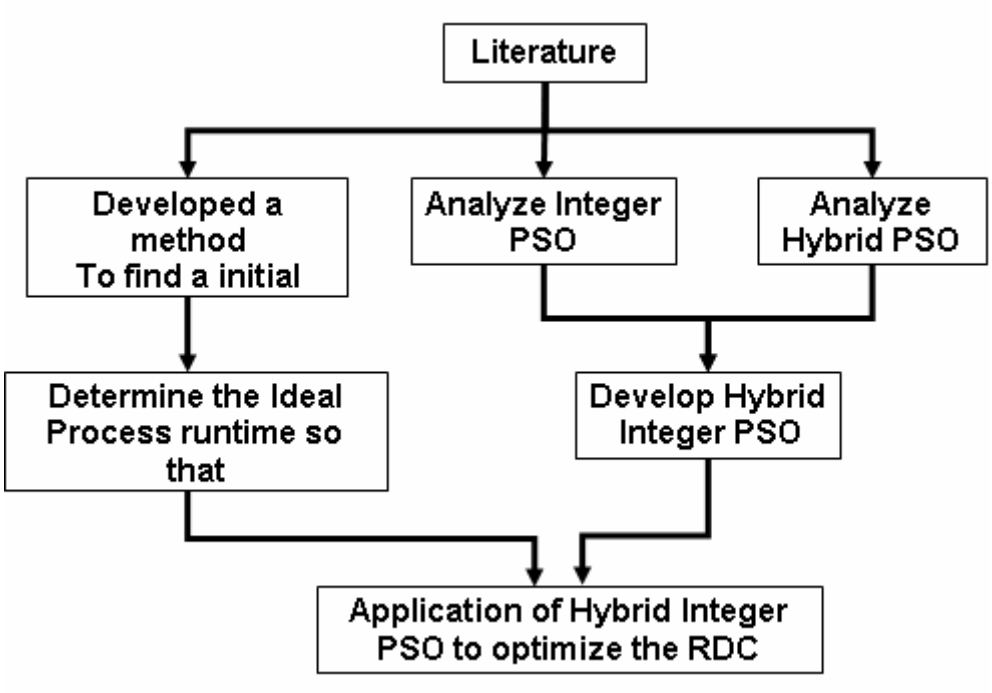

Figure 1.1 Research Methodology

The amount of time required by the system to come back to steady state depends upon the order of the system. The higher the order the larger is the time taken by the system to reach the steady state. A considerable time should be provided so that any system with in the specified range can attain steady state after the initial values are used. Since we did not use any criteria to determine the relation between the system and the time required by

it to reach steady state, the system was operated for a fixed time long enough so that any design based on order of the system can attain steady state.

Although there is no guarantee that the system will not blow up it was noticed that process operation time as large as 16 hours should be sufficient for any system which can give us desired results to accommodate steady state. There after the Hybrid-Integer PSO model developed is used as an optimization tool upon the process so as to find the optimal design based on our requirements.

1.5 Significance

Many chemical industries and petroleum refineries use reactive distillation columns for separation methods. A huge amount of installation cost is required to setup such a facility and maintaining it is no exception especially when one of the inputs fed is fuel (oil), mere mention of the word oil is sufficient to realize how beneficial it is to save even 1% on fuel expenditures when the consumption is in barrels per day. The rise in cost of fuel day by day and the ever increasing demand in the production sector of petro-chemicals necessitate proper efficiency of fuel; one of the requirements in getting the maximum efficiency out of fuel is the proper design and least consumption to get the maximum output. This work is another step towards optimizing a process so as to conserve the fuel intake and thereby save capital.

1.6 Contribution

- In this work Hybrid Integer Particle Swarm Optimization (HI-PSO) algorithm is developed to solve Integer Problems which guarantees faster convergence and avoids Local Minima Stagnation.
- Implementation of HI-PSO which minimizes fuel consumption in Ideal Reactive Distillation Column to obtain an optimal design

CHAPTER 2

PARTICAL SWAM OPTMIZATION

2.1 The Basic Particle Swarm

The concept of function-optimization by means of a particle swarm was introduced by James Kennedy, Russel and Eberhart in an IEEE neural network conference paper from 1995 [12]. The method was discovered through simulation of a simplified social model. In this model, each particle position can be thought of as a state of mind as a particular setting of the abstract variables that describe our beliefs and attitudes. Movement of particles in this model then corresponds to the concept of change of mind. Humans adjust their beliefs to each other; we evaluate stimuli from the environment, compare it to ourselves, and finally imitate the stimuli. These three important properties of human social behavior - evaluation, comparison, and imitation - is the main inspiration for the particle swarm, and the particle swarm utilizes these concepts in adapting to environmental changes and solving complex and hard problems [11].

Besides being a model of the human social behavior, the particle swarm (as noted by Kennedy [12] and discussed in chapter 4) is closely related to swarm intelligence. In the particle swarm, there is no central control no one gives orders. Each particle is a simple

agent acting upon local information. Yet, the swarm as a whole is able to perform tasks, whose degree of complexity is well beyond the capabilities of the individual. The particle swarm shows signs of self-organization: The interactions among the low-level components (particles) result in complex structures at the global level (swarm) making it possible for it to perform optimization of functions.

According to Kennedy, five basic principles define swarm intelligence [11]. First is the proximity principle: the swarm should be able to carry out simple space and time computations. Second is the quality principle: the swarm should be able to respond to quality factors in the environment. Third is the principle of diverse response: the swarm should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the swarm should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the swarm must be able to change behavior mote when it is worth the computational price. Note that principles four and five are the opposite sides of the same coin.

The particle swarm seems to adhere to all five principles [12]. Conclusively, the particle swarm strategies should be thought of a swarm intelligent system. Further, the particle swarm has roots in artificial life and in evolutionary computation (EC). Indeed, each particle in the swarm is a simple agent that acts in an environment according to a rule set

that takes the state of the environment and the agents into account when deciding what action to choose, like many artificial life applications.

The connection to evolutionary computation is obvious. The swarm consists of a population of individuals that represent solutions to the optimization problem, we would like to solve. Through an iterative and probabilistic modification of these solutions, we search for an optimal solution. Describing the particle swarm in these EC-terms, the leap to the evolutionary algorithm is little. The difference between the EA and the PSO with a simple of view only, how we change the population/swarm from one iteration to the next; in the EA, genetic operators like selection, crossover and mutation, are used, whereas the particle in the PSO are modified according to two update formulas that we shall be familiar with in section 2.1.1

Conceptually, there are, however some differences between the PSO and EA: In PSO Particles stay alive and inhabit the search space during the run, whereas in EA the individuals are replaced each generation. Furthermore, the objective is reached through cooperative search in particle swarms rather than competitive search as in EA.

2.1.1 The Algorithm

Next, we present the particle swarm algorithm for optimization of continuous and real-valued functions in the *n*-dimensional space, $\mathbb{R}^n$. The PSO is a population-based search-algorithm; the population is called a swarm *S*. The swarm consists of a number of particles that move around in the search space *S*. A neighborhood relation *N* is defined on the swarm. *N* determines for all particles p_i and p_j whether they are neighbors or not, and we can thus for each particle *p* assign a neighborhood, *N*(*p*), containing all neighbors of *p*. A fitness function *f* must be defined to compare candidate solutions in the search space *S*, which is a subset of $\mathbb{R}^n$, and map into the real numbers, i.e.: $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$. In fact, the PSO only compares fitness, so an ordinal fitness function would suffice. Each particle *p* has two state variables:

- Its current position: $\vec{x}(t)$,
- Its current velocity: $\vec{v}(t)$,

As well as a small memory containing:

- Its best position: $\vec{p}(t)$, and
- The best $\vec{p}(t)$ of all $p \in N(p)$: $\vec{g}(t)$,

Where $\vec{p}(t)$, $\vec{g}(t)$, $\vec{x}(t)$ and $\vec{v}(t)$ are n-dimensional vectors.

Particle Swarm Optimization consists of three parameters basically: v_{max}, which restricts every coordinate of $\vec{v}(t)$ within the range [-v_{max} to v_{max}] and ϕ_1 and ϕ_2 that determine the influence of $\vec{p}(t)$ and $\vec{g}(t)$ in the velocity update formula.

The swarm is initialized at time t = 0 by placing the particles randomly and uniformly distributed in S and assigning a random and uniformly chosen velocity vector $\vec{v}(0)$ from V^n. moreover, we set $\vec{p}(0)=\vec{g}(0)=\vec{x}(0)$.

The iterative optimization process starts after this initialization. The expressions for the particle positions and velocities in the next time step are given by these recursive equations:

$$\vec{v}(t+1)=\vec{v}(t)+\phi_1\left(\vec{p}(t)-\vec{x}(t)\right)+\phi_2\left(\vec{g}(t)-\vec{x}(t)\right) \tag{2.1}$$

$$\vec{x}(t+1)=\vec{x}(t)+\vec{v}(t+1) \tag{2.2}$$

The position of a particle at time t+1 is calculated as a sum of it old position $\vec{x}(t)$ and current velocity $\vec{v}(t+1)$. Additionally, the velocity $\vec{v}(t+1)$ is updated as a sum of the particle's old velocity $\vec{v}(t)$, its own *cognitive learning* part $\phi_1\left(\vec{p}(t)-\vec{x}(t)\right)$ and *social learning* part $\phi_2\left(\vec{g}(t)-\vec{x}(t)\right)$.

After having calculated the velocities and position for the next time step t+1, the first iteration of the algorithm is completed. Typically, this process is iterated for a certain number of time steps, or until some acceptable solution has been found by the algorithm. Here we present the pseudo-code for the PSO algorithm. During the search, the Particles exchange information about their positions and fitness values. This communication results in, that the swarms learn and refine its knowledge about the search, and move towards the good search space areas. This is analogous to flocks of birds flying and searching for food, to social insects such as bees and ants when foraging or nesting, and to humans that affect the minds of each other by interacting socially.

Program Particle Swarm Optimization Algorithm

Set t = 0;

Initialize ϕ_1, ϕ_2, V_{max} *and define N;*

$\forall p \in S$: *Initialize* $\vec{x}(t), \vec{v}(t), \vec{p}(t), \vec{g}(t)$ *as described;*

While {Min. is not reached or Iterations not exhausted

$\forall p \in S$: *Calculate* $\vec{v}(t+1)$ *and* $\vec{x}(t+1)$ *using 1 and 2*

$\forall p \in S$: *Update* $\vec{p}(t+1)$ *with* $\vec{x}(t+1)$ *if* $f\left(\vec{x}(t+1)\right)$ *is better than* $f\left(\vec{x}(t)\right)$

$\forall p \in S$: *Update* $\vec{g}(t+1)$ *with* $\vec{p}(t+1)$ *in N(p)*

}

These analogies in nature of refinement of knowledge by cooperation have been the inspiration for the PSO. Therefore, as an emergent result of the two simple equations (1) and (2) above, the swarm as a whole will identify and approach the good areas of the search space in a self-organized structure based on comparison to and imitation of each other.

On the algorithmic level, the main strength of the PSO is its fast convergence, which compares favorable to many EA implementations. However, it has three major eaknesses:

- It cannot dynamically adjust its velocities when fine-tuning a found optimum, and hence the convergence rate decreases dramatically in the close vicinity of optima [13].
- On hard problem for instance with many optima, its fast convergence rate often results in premature convergence [14].
- The number of PSO parameters to tune is critically big [14]

Next we describe extensions that deal with the first two issues, and in the following section we discuss the matter of parameter selection in the PSO.

2.2 Extensions to the Particle Swarm

2.2.1 Controlling the Convergence

The PSO presented in section 2.1 optimizes continuous, real-valued functions and is able to find an optimum if not the global optimum, then at least a local one. However, if it was not for the introduced parameter v_{max}, the swarm would not converge at all. Instead it would diverge in sinus-like waves of increasing amplitudes without being able to optimize at all [11]. It is obvious that v_{max} is necessary for the PSO to actually perform optimization. Even, when v_{max} is applied and the swarm may seem to settle on an optimum, the particles in the PSO do actually not converge towards a point. Using v_{max} imposes a maximum velocity step size on the particles, and this avoids divergence as we find it in the situation without v_{max}. Now the particles are kept within a distance roughly equal to v_{max} from the found optimum-point $\vec{g}(t)$, but they will, in general, not get closer to it as the optimization progresses. A particle can of course be fortunate to hit a position very close (or equal to) the optimum, but since the search area is not narrowed in over time, the evaluations are allocated more or less uniformly over the interval $[\vec{g}(t) - v_{\max}, \vec{g}(t) + v_{\max}]$. Hence, if we want to investigate the near neighborhood of $\vec{g}(t)$ for purposes of fine-tuning, the v_{max} approach is obviously not efficient. In conclusion, the v_{max} approach avoids divergence is inefficient near optima compared to other search techniques such as genetic algorithms that often use annealing mutation schemes to obtain better fine-tuning [13].

This problem consideration has resulted in two proposed changes to the PSO that each solve this problem. Today these changes have become an integrated part of the PSO model, because of the resulting performance improvements.

The first proposed model, the inertia-weight model, by Eberhart and Shi [18] multiplies the velocity of the current time-step *t* with a factor called the inertia weight, ω, in the calculation of the new velocity at $t + 1$:

$$\omega * \vec{v}(t) + \phi_1 \left(\vec{p}(t) - \vec{x}(t) \right) + \phi_2 \left(\vec{g}(t) - \vec{x}(t) \right) \tag{2.3}$$

Where $\omega \in [0,1]$ is to enforce convergence. The reducing factor ω is only multiplied with $\vec{v}(t)$.

The second model by Maurice-Clerc introduces a constriction-factor χ [15]. The intention is to *constrict* the velocity by multiplying it with χ before updating $\vec{x}(t)$. The velocity update formula now looks like:

$$\vec{v}(t+1) = \chi \left(\vec{v}(t) + \phi_1 \left(\vec{p}(t) - \vec{x}(t) \right) + \phi_2 \left(\vec{g}(t) - \vec{x}(t) \right) \right) \tag{2.4}$$

Where $\chi \in [0,1]$ This strategy obviously reduces $\vec{v}(t)$ at every time-step compared to the original velocity update formula, and by setting χ sufficiently low, we can assure convergence [15].

It is easy to realize that the two modifications of the velocity update formula are in effect equivalent: The constriction-factor can mimic the inertia-weight, and vice versa: if we have a setting of $(\chi, \phi_1, \phi_2) = (x, y, z)$ in the constriction model, we obtain an equivalent setting in the inertia-weight model by setting $(\omega, \phi_1, \phi_2) = (x, xy, xz)$. Thus, to transfer settings between the two models it suffices to multiply (or respectively divide) ϕ_1 and ϕ_2 with the value of χ.

2.2.2 Avoiding Premature Convergence

In this section we present extensions to the PSO model that is concerned with the problem of premature convergence to sub-optimal solutions. They do not only manipulate the velocity update formula, but build up a genuinely new model on the swarm level. The PSO model converges by nature rather quickly (i.e. the diversity in the swarm decreases quickly). This is exact, what is wanted when the problem in question is easy e.g. is a uni-modal problem. On such easy problems, the convergence should be as fast as possible, because there is no risk of being trapped on a sub-optimal solution. However, when we face more difficult, multi-modal problems, then too fast convergence often becomes inadequate and unwanted. When there are many different local optima, we must spend more time on investigating different solution areas before converging simply to avoid getting stuck in a sub-optimal area; i.e. to avoid premature convergence.

This topic has been addressed in several papers. The hybrid-PSO model by Løvbjerg and colleagues uses breeding between particles and sub-populations [16]. Since breeding is a core element in the GA, the authors hypothesize that a PSO with breeding might reach a better optimum. Breeding is implemented through arithmetic crossover as known from GAs; a parameter p_b is introduced to control the probability of breeding. Further, offspring replace parents, and the population is divided into sub-populations to avoid premature convergence. Finally, a parameter p_{sb} was added to control the probability of breeding inside sub-populations. Based on their research, the authors conclude that marginally faster convergence is obtained with the hybrid-PSO, and that the best found values are better on multi-modal problems but worse on uni-modal. This model has been explained in upcoming section and this forms the basis of our Modified Algorithm.

2.2.3 Speeding up Convergence

The issue of speeding up convergence towards optimum has not been thoroughly investigated yet. The primary reason for this is that precisely the convergence speed of the swarm is already an inherent force of the PSO construction as discussed by Angeline [13]. Hence, the PSO converges to the fitness optimum quickly on easy problems, but is also more vulnerable to premature convergence on multi-modal problems. With ω and χ added to the algorithm, the rapid convergence even lasts throughout the whole optimization process so fine-tuning of solutions is efficient. Obtaining fast convergence

has therefore in much research therefore been reduced to tuning ω and χ, rather than creating mechanisms that changes the fundamental behavior of the particles.

Kennedy and colleagues present a model that tries to improve particles trajectories [11]. He first approximates a number *C* of cluster-centers. The idea is that these centers might be nearer to the optimum (around which the particles are swarming), than the particles themselves, and thus substituting these centers for $\vec{p}(t)$ and $\vec{g}(t)$ might improve their search. This approach tries to improve the convergence within sub-clusters, which leads to faster convergence towards the fitness optimum. Since the convergence rate in only increased within clusters (that will converge anyway), this model has the nice property that it tries to improve the convergence rate towards good solutions without increasing the risk of converging on sub-optimal solutions. Substitutions of $\vec{p}(t)$ only, of $\vec{g}(t)$ only, and of both are tested. The author concluded that average performance per fixed number of iterations can be improved by substituting, but the results are only preliminary.

2.3 Parameter Selection

With the introduction of the inertia weight and the constriction factor as fundamental concepts of the PSO, there is quite a number of parameters to consider in the PSO-

algorithm: $\omega, \chi, \phi_1, \phi_2$ and v_{max}: Thus, at this point it might be a good idea to take a look at how the parameters should be controlled in the PSO model.

The parameter-settings of the PSO determine how it optimizes the search-space. For instance, one can apply a general setting that gives reasonable results on most problems, but seldom is very optimal. A useful setting for a general search is to set $\phi_1 = \phi_2 = 2$ and the inertia weight ω = 0:8. The value of v_{max} depends on the size of S (and properties of the function being optimized), but for practical use v_{max} is often set to approximately. 10% of the average dimension size of S.

But these settings cannot be used on many problems with optimal success; hence we must have knowledge of the effects of the different settings, so we can pick a suitable setting from problem to problem. For instance, if the problem for which we are optimizing has a uni-modal fitness-landscape, we most likely want the PSO to act as a hill-climber, and we can set the parameters in a specific way to support an efficient hill-climbing behavior. In contrast, in other cases where the fitness landscape has many peaks, we can set the parameters differently to adapt the behavior to multi-modal problem-domains.

2.3.1 The Control Parameters ϕ_1 and ϕ_2

There are two important facts to consider, when setting ϕ_1 and ϕ_2 .The first fact is that the relation between the two values decides the point of attraction, which is given by:

$$\frac{\phi_1 \vec{p}(t) + \phi_2 \vec{g}(t)}{\phi_1 + \phi_2}$$

If $\phi_1 >> \phi_2$, the particle *p* will be much more attracted to the best found position by itself, $\vec{p}(t)$, rather than the best position found by the neighborhood $\vec{g}(t)$, and vice versa if $\phi_1 < \phi_2$.

The extreme case $\phi_2 = 0$ converts all particles to independent hill-climbers since the social learning part $\phi_2\left(\vec{g}(t) - \vec{x}(t)\right)$ is 0. The iterated hill-climber finds the best point in the neighborhood by replacing the current point, if a better is found. This is repeated until a local optimum is reached, and the whole process can in addition be repeated with new starting points as many times as one wishes. Similarly, the PSO with setting $\phi_2 = 0$ swarms about the point $\vec{p}(t)$, and searches a neighborhood, whose size is indirectly defined by v_{max}. If the particle finds a better solution (and this will eventually happen if $\vec{p}(t)$ is not already equal to the optimum) $\vec{p}(t)$ will be updated, and the particle will start swarming around the updated $\vec{p}(t)$. Conclusively, the particle (precisely as the hill-

climber) moves uphill, and this process continues iteratively until an optimum is found, in accordance with the behavior of a hill-climber). Conversely, the hill-climber particle in the PSO does not know exactly when the optimum is found, because it does not systematically (exhaustively) search the neighborhood.

In the other extreme case, where $\phi_1 = 0$, the particle own cognitive learning part $\phi_1\left(\vec{p}(t) - \vec{x}(t)\right)$ is 0, and the whole swarm is attracted to one single point only, namely $\vec{g}(t)$. Essentially, the swarm now turns into one big hill-climber as described above; all particles swarm around $\vec{g}(t)$, and moves with each update of $\vec{g}(t)$. The neighborhood is searched in parallel by all the particles simultaneously, and resulting in one parallel, stochastic hill-climber.

If $\phi_1 = \phi_2$, each particle will be attracted to the average of $\vec{p}(t)$ and $\vec{g}(t)$. Since ϕ_1 expresses how much the particle trusts its own past experience, it is called the cognitive parameter, and since ϕ_2 expresses how much it trusts the swarm, it is called the social parameter. Most implementations use a setting with ϕ_1 roughly equal to ϕ_2. However, Carlisle and Dozier [17] reported good results with $\phi_1 = 2{:}8$ and $\phi_2 = 1{:}3$ [17].

The second fact to consider when setting the control variables is the magnitudes of ϕ_1 and ϕ_2. The higher ϕ_1 and ϕ_2 the more acceleration the particles can obtain. In fact, at any time a particle's acceleration is given by the term: $\phi_1\left(\vec{p}(t)-\vec{x}(t)\right)+\phi_2\left(\vec{g}(t)-\vec{x}(t)\right)$ (Without inertia-weight or constriction). Thus, setting the control variables high, enables the swarm to react rapidly to changes in the search, whereas if they are set low, the particles will react slowly and move in waves of huge-magnitude and low-frequency. They generally move farther away from the point of attraction $\frac{\phi_1\vec{p}(t)+\phi_2\vec{g}(t)}{\phi_1+\phi_2}$ and will not change direction as often as with the control variables set high.

2.3.2 The Inertia Weight ω

The inertia weight ω controls the momentum of the particle: If ω << 1, only little momentum is preserved from the previous time-step; thus quick changes of direction are possible with this setting. The concept of velocity is completely lost if ω = 0, and the particle then moves in each step without knowledge of the past velocity. On the other hand, if ω is high (> 1) we get the same effect as when ϕ_1 and ϕ_2 are low: Particles can hardly change their direction and turn around, which of course implies a larger area of exploration as well as a reluctance against convergence towards optimum. Setting ω > 1 must be done with care, since velocities are further biased for an exponential growth.

This setting is rarely seen in PSO implementation, and always together with v_{max}. In short, high settings near 1 facilitate global search, and lower settings in the range [0:2; 0:5] facilitate rapid local search.

R. Eberhart and Y. Shi have studied ω in several papers and found that when v_{max} is not small ($\geq$ 3), an inertia-weight of 0.8 is a good choice [18]. Although this statement is solely based on a single test function, the Schaffer function, this setting actually is a good choice in many cases. The authors have also applied an annealing scheme for the ω-setting of the PSO, where ω decreases from ω = 0.9 to ω = 0.4 over the whole run [19]. They compared their annealing scheme results to results with ω = 1 obtained by Angeline [13], and conclude a significant performance improvement on the four tested functions. The decreasing ω-strategy is a near-optimal setting for many problems, since it allows the swarm to explore the search-space in the beginning of the run, and still manages to shift towards a local search when fine-tuning is needed.

2.3.3 The Constriction Factor χ

The constriction factor model has as mentioned in section 2.2.1 the same effect as ω, except that it also scales the contributions from $\vec{p}(t)$ and $\vec{g}(t)$ with χ. Thus, any regulation of the ability to change direction (i.e. the relationship between $\vec{v}(t)$ and

$\phi_1\left(\vec{p}(t)-\vec{x}(t)\right)+\phi_2\left(\vec{g}(t)-\vec{x}(t)\right)$ must be done with ϕ_1 and ϕ_2 with this model. However, basically χ acts as ω: Low values facilitates rapid convergence and little exploration where as high values gives slow convergence and much exploration.

One of the few theoretical founded contributions to particle swarm research comes from the mathematician Maurice Clerc, who has proposed the constriction factor [15]. He has studied the particle swarm system by means of second order differential equations. In doing so, it is possible to determine under which conditions the swarm will converge. However, we must emphasize that this analysis did not model a swarm of particles, but only one single deterministic particle in a one-dimensional space. Further, the author assumed a static $\vec{p}(t)$ and $\vec{g}(t)$. In spite of these simplifications, it is worth looking at one particular outcome of the analysis. In the constriction model we can set χ as a function of ϕ_1 and ϕ_2, so that convergence is ensured even without v_{max}. An additional parameter *k*, which controls the convergence speed of the particles to the point of attraction, is introduced instead of $\hat{A}$ (see eq. 5).

$$\chi = \frac{2k}{\left|2-\phi-\sqrt{\phi(\phi-4)}\right|} \text{ where } \phi=\phi_1+\phi_2 \geq 4, k \in [0,1] \tag{2.5}$$

The supposed advantage of this shift from χ to *k*, is that *k* more clearly and reliably can control swarm behavior: If *k* is close to 0, we get fast convergence (almost hill-climbing behavior), and if *k* is near 1 we get the slowest possible convergence with a high degree of exploration, which is desired for strongly multimodal problems.

2.3.4 The Maximum Velocity v_{max}

Originally, v_{max} was introduced to avoid explosion and divergence. With χ or ω in the update formula, v_{max} to some degree has become unnecessary; at least convergence can be assured without it [15]. Thus, some researchers simply do not use v_{max}. In spite of this fact, the maximum velocity limitation can still improve the search. For instance, if a particle is positioned in one end of the search-space and $\vec{p}(t)$ and $\vec{g}(t)$ in the other end, the particle will be able to obtain a velocity of four times i.e. If the distance, *d*, from $\vec{x}(t)$ to $\vec{p}(t)$ and $\vec{g}(t)$ is approximately the length of the search space, and we assume $\phi_1 \approx \phi_2 \approx 2$, then from eq. (2.1) we get that $v(t+1) \approx 4d$ i.e. four times the size of the search space, which obviously is nonsense. Hence, after moving to the other end of the search-space, in what probably is one timestep, the particle will now have to spend time (and evaluations) decelerating its velocity before being able to turn around. In this period the search is more or less locked up, and still the PSO uses CPU-time on evaluating the

same boundary solutions again and again. Alternatively, it could have approached the point of attraction.

$$\frac{\phi_1 \vec{p}(t) + \phi_2 \vec{g}(t)}{\phi_1 + \phi_2} \tag{2.6}$$

In a controlled and more efficient manner with a suitable v_{max} applied, thus gaining useful information about the fitness landscape between the two corners by sampling different solutions on its way across the search space.

2.3.5 The Neighborhood Topology

Another important factor in PSO is the neighborhood topology. The PSO presented in 2.1.1, implicitly uses a so-called fully connected neighborhood topology (or *gbest*). Every particle is neighbor of every other particle. In a paper by Kennedy, other topologies are described as well [14]:

- The k-best topology, which connects every particle to its *k* nearest particles in the topological space. With *k* = 2, this becomes the circle topology (and with *k* = swarmsize-1 it becomes a *gbest* topology).
- The wheel topology, in which the only connections are from one central particle to the others (see figure). In addition, one could magine a huge number of other topologies.

In the mentioned paper, Kennedy investigated the three described topologies, and concluded that the topology has an effect on the search results. Nevertheless, these effects are hard to find when analyzing the obtained results.

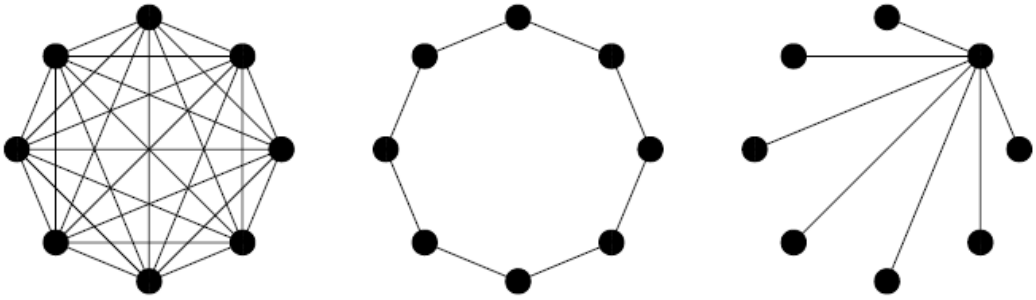

Figure 2.1: a) fully connected b) k-best with $k = 2$, and c) twheel topology

2.4 Constrained Problems

In section 2.1 we presented the PSO, and defined it to operate on a subset S of $\mathbb{R}^n$

Rn. One type of subset is the hyper-cube (HC), where each dimension is bounded by an upper value (u_i) and a lower value (l_i): $HC = \{x \in \mathbb{R}^n \mid \forall i : l_i < x_i < u_i\}$

The hyper-cube is the domain-type used in many test-problems, because it is simple and the constraint-handling should not be an issue. Nonetheless, below we show that even on this simplest type of constraint, the PSO can fail.

Suppose we initialize the swarm inside a hyper-cube with a uni-modal fitness function having its optimum in the center of the cube. One particle might be placed near to the

center in all dimensions except in one dimension, where it is very near the boundary. We call this dimension b. Being relatively close to the global optimum, it might get the best fitness value of all particles. Further, let us assume that its velocity $v_b(t)$ points towards the border in this dimension b such that $x_b(t)$ crosses the border in the next timestep. Following the standard PSO implementation of domain-border violations, we adjust the coordinate-value to lie on the border: $x_b(t) = u_b$ (or l_b).

As long as this particle's fitness is not lower than any of the others, two things will happen:

- $x_b t$ will remain equal to u_b, because $p_b(t)$ and $g_b(t)$ also are equal to $x_b(t)$, and thus the velocity $v_b(t)$ will continue to point outwards of the search-space.
- The b'th coordinate of the other particles will be attracted to $x_b(t)$, and sooner or later they will end up with the value u_b. If this happens before a new $gb(t)$ is found by another particle, the b'th coordinate of all the particles will be stuck at ub forever.

This problem can be avoided by simply allowing the particles to move outside the search space S, but without evaluating outside the search space. As a particle cannot improve its fitness being outside, $\vec{p}(t)$ and $\vec{g}(t)$ are always updated inside S, and after some time the particle will enter S again because of the attraction to exactly $\vec{p}(t)$ and $\vec{g}(t)$. This scheme can be improved even further.

To conclude particle Swarm optimization in its basic form although is very simple but may suffer from inabilities to reach the global minima. Few modifications need to be done, although these modifications like, inertia weight, neighborhood, constriction factor and the V_{max} operator have become an intrinsic part of the algorithm, there is still room for development when coming to specific problems.

CHAPTER 3

HYBRID INTERGER PSO

In this chapter we develop the Hybrid-Integer PSO model based on the existing extensions to particle Swarm Optimization, The integer PSO and the Hybrid-PSO. In section 3.1.1 we discuss in detail the Hybrid particle Swarm optimization, Followed by integer programming of Particle Swarm Optimization in section 3.1.2. In section 3.2 the shortcoming of both the algorithms are discussed and the need to amalgamate them so as to bring a more hybridized version of Integer PSO is put forward. The chapter is concluded by testing some standard integer test problems.

3.1 Background

3.1.1 Hybrid PSO

Hybrid Particle Swarm Optimizers is combining the idea of the particle swarm with concepts from Evolutionary Algorithms. The hybrid PSO combines the traditional velocity and position update rules with the ideas of breeding and subpopulations. PSO with breeding strategies have the potential to achieve faster convergence and the potential to find a better solution. Both Eberhart [11] and Angeline[13] conclude that hybrid models of the standard GA and the PSO could lead to further advances. Lovberg et al.

present such a hybrid model [16]. The model incorporates one major aspect of the standard GA into the PSO, the reproduction. In here we will refer to the used 'reproduction' and 'recombination' of genes only as "breeding". Breeding is one of the core elements that make the standard GA, a powerful algorithm. In addition to breeding Lovberg et al. introduce a hybrid with both breeding and subpopulations. Subpopulations have previously been introduced to standard GA models mainly to prevent premature convergence to suboptimal points [37]. The motivation for this extension was that the PSO models, including the hybrid PSO with breeding, may also reach suboptimal solutions. Breeding between particles in different sub-populations was also added as an interaction mechanism between subpopulations.

The particles have no neighborhood restrictions, meaning that each particle can affect all other particles. This neighborhood is of type star (fully connected network), which have been shown to be a good neighborhood type in [14]. The structure of the hybrid model is illustrated.

```
Start
        Initialize PSO
        while ( Termination Criteria not Satisfied) do
                Start PSO
                Evaluate
                Calculate new velocity vectors
                Update Positions
                Breed
        End}
end
```

The breeding is done by first determining which of the particles that should breed. This is done by iterating through all the particles and, with probability *pb* (breeding probability), mark a given particle for breeding. Note that the fitness is not used when selecting particles for breeding. From the pool of marked particles we now select two random particles for breeding. This is done until the pool of marked particles is empty. The parent particles are replaced by their offspring particles, thereby keeping the population size fixed.

The position of the offspring is found for each dimension by arithmetic crossover on the position of the parents, i.e.,

$$Child_1(x_i)=p_i*parent_1(x_i)+ (1-p_i)*parent_2(x_i) \tag{3.4}$$

$$Child_2(x_i)=p_i*parent_2(x_i)+ (1-p_i)*parent_1(x_i) \tag{3.5}$$

Where p_i is a uniformly distributed random value between 0 and 1. The velocity vector of the offspring is calculated as the sum of the velocity vectors of the parents normalized to the original length of each parent velocity vector.

$$child_1(\vec{v}) = \frac{parent_1(\vec{v}) + parent_2(\vec{v})}{\left|parent_1(\vec{v}) + parent_2(\vec{v})\right|}\left|parent_1(\vec{v})\right| \tag{3.6}$$

$$child_2(\vec{v}) = \frac{parent_1(\vec{v}) + parent_2(\vec{v})}{\left|parent_1(\vec{v}) + parent_2(\vec{v})\right|}\left|parent_2(\vec{v})\right| \tag{3.7}$$

The arithmetic crossover of positions and velocity vectors used were empirically tested to be the most promising. The arithmetic crossover of positions in the search space is one of the most commonly used crossover methods with standard real valued GA's, placing the offspring within the hypercube spanned by the parent particles. The main motivation behind the crossover is that offspring particles benefit from both parents. In theory this allows good examination of the search space between particles. Having two particles on different suboptimal peaks breed could result in an escape from a local optimum, and thus aid in achieving a better one. We used the same idea for the crossover of the velocity vector. Adding the velocity vectors of the parents results in the velocity vector of the offspring. Thus each parent affects the direction of each offspring velocity vector equally. In order to control that the offspring velocity was not getting too fast or too slow, the offspring velocity vector is normalized to the length of the velocity vector of one of the parent particles. Finally, the starting position of a new offspring particle is used as the initial value for this particle's best found optimum $\overrightarrow{p_i}(t)$.

The motivation for introducing subpopulations is to restrict the gene flow (keeping the diversity) and thereby attempt to evade suboptimal convergence. The subpopulation hybrid PSO model is an extension of the just described breeding hybrid PSO model. In this new model the particles are divided into a number of subpopulations. The purpose of the subpopulations is that each subpopulation has its own unique best known optimum. The velocity vector of a particle is updated as before except that the best known position

($\overrightarrow{g_i}(t)$ in the formula) now refers to the best known position within the subpopulation that the particle belongs to. In terms of the neighborhood topology suggested by Kennedy in [14], each subpopulation has its own neighborhood. The only interaction between subpopulations is if parents from different subpopulations breed. Breeding is now possible both within a subpopulation but also between different subpopulations. An extra parameter called probability of same subpopulation breeding (p_{sb}) determines whether a given particle selected for breeding is to breed within the same subpopulation (probability p_{sb}), or with a particle from another subpopulation (probability 1- p_{sb}). Replacing each parent with an offspring particle ensures a constant subpopulation size.

As an example we take a function $F(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$ which is generally called Rastrigins Function. It has a global minimum of zero and has the bounds in the range $x \in [-5.12, 5.12]$, $F(x) = 0$. The PSO parameters were as follows:

The maximum number of allowed function evaluations was set to 10000 (500 iterations); the desired accuracy was 100%; the constriction factor was set equal to 1; the inertia weight ω was gradually decreased from 0.7 towards 0.4; c1 = c2 = 2; and V_{max} = 100 and the size of the Swarm were selected to be equal to 20.

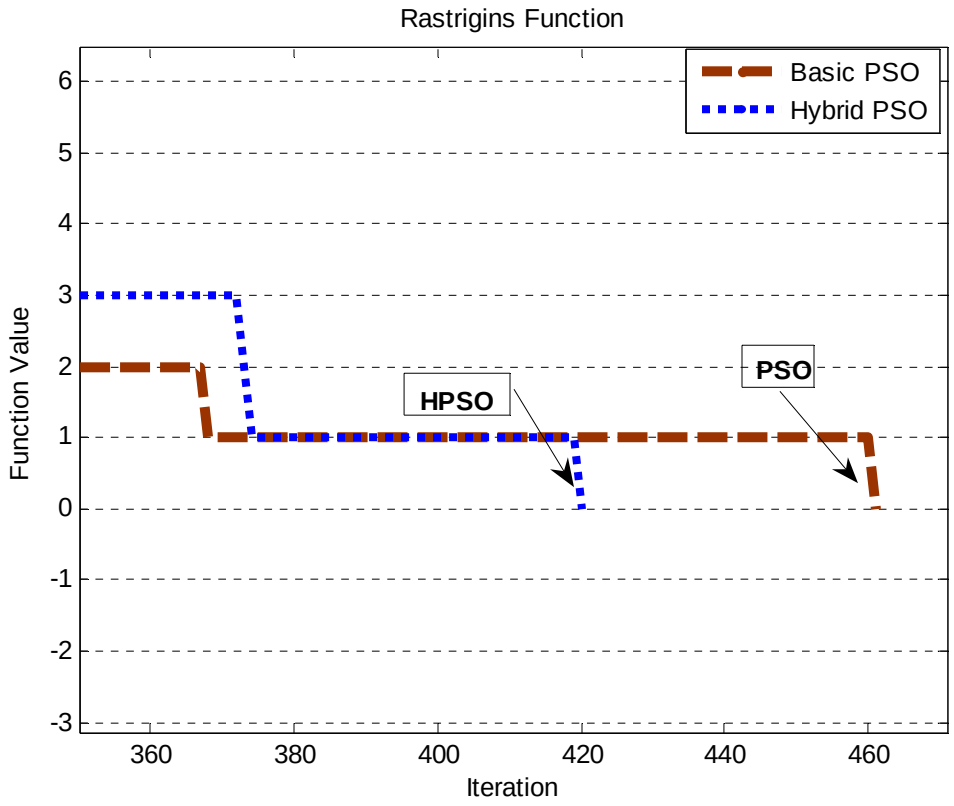

Figure 3.1 Basic PSO vs Hybrid PSO for Rastrigins Function

From the results in Figure 3.1 its quite clear that Hybrid version of PSO is superior to the Basic Version. Note that the parameters of PSO were maintained same for both the version.

3.1.2 Integer PSO

A wide variety of problems can be represented as discrete optimization models. An important area of application concerns the efficient management of a limited number of resources so as to increase productivity and/or profit. Such applications are encountered in Operational Research problems such as goods distribution, production scheduling, and machine sequencing. There are applications in mathematics to the subjects of graph theory and logic [27].

Statistical applications include problems of data analysis and reliability. Recent scientific applications involve problems in molecular biology, high energy physics and x-ray crystallography. A political application concerns the division of a region into election districts [27]. Capital budgeting, portfolio analysis, network and VLSI circuit design, as well as automated production systems are some more applications in which Integer Programming problems are met [27].

Yet another, recent, and promising application is the training of neural networks with integer weights, where the activation function and weight values are confined in a narrow band of integers. Such neural networks are better suited for hardware implementations compared to real weight ones [28].

The Unconstrained Integer Programming problem can be defined as

$$\min_{x} f(x), x \in S \subseteq \mathbb{Z}^n \tag{3.1}$$

Where Z is the set of integers, and S is a not necessarily bounded set, which is considered as the feasible region. Maximization of Integer Programming problems is very common in the literature, but we will consider only the minimization case, since a maximization problem can be easily transformed to a minimization problem and vice versa. The problem defined in Eq. (3.1) is often called “All Integer Programming Problem", since all the variables are integers, in contrast to the “Mixed Integer Programming Problem", where some of the variables are real.

Optimization techniques developed for real search spaces can be applied on Integer Programming problems and determine the optimum solution by rounding off the real optimum values to the nearest integer [27], [29].

Evolutionary and Swarm Intelligence algorithms are stochastic optimization methods that involve algorithmic mechanisms similar to natural evolution and social behavior respectively. They can cope with problems that involve discontinuous objective functions and disjoint search spaces [30], [11], [32]. Genetic Algorithms (GA), Evolution Strategies (ES), and the Particle Swarm Optimizer (PSO) are the most common paradigms of such methods. GA and ES draw from principles of natural evolution which are regarded as rules in the optimization process. On the other hand, PSO is based on

simulation of social behavior. Early approaches in the direction of Evolutionary Algorithms for Integer Programming are reported in [34], [35]. In GA, the potential solutions are encoded in binary bit strings. Since the integer search space, of the problem defined in Eq. (3.1), is potentially not bounded, the representation of a solution using a fixed length binary string is not feasible [33]. Alternatively, ES can be used, by embedding the search space Z^n into R^n and truncating the real values to integers.

However, this approach is not always efficient due to the existence of features of ES, which contribute to the detection of real valued minima with arbitrary accuracy. These features are not always needed in integer spaces, since the smallest distance of two points, in 1-norm, is equal to 1 [33].

In Integer-PSO the velocity of the particles and the Swarm positions are updated as per the standard equations mentioned in the previous chapter. Integer Programming test problem was selected to investigate the performance of the PSO method. Each particle of the swarm was truncated to the closest integer, after the determination of its new velocity using

$$\vec{v}(t+1) = \omega * \vec{v}(t) + \phi_1\left(\vec{p}(t) - \vec{x}(t)\right) + \phi_2\left(\vec{g}(t) - \vec{x}(t)\right) + \phi_3\left(\vec{n}(t) - \vec{x}(t)\right) \tag{3.2}$$

And positions using

$$\vec{x}(t+1) = \vec{x}(t) + \chi\vec{v}(t+1) \tag{3.3}$$

As an example $F(x) = -(15\ 27\ 36\ 18\ 12)x + x^T \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix}$

The range of x is $x \in [-100,100]$ and the solutions is $F(x) = -737$ with $x = (0,11,22,16,6)\ and\ x = (0,12,23,17,6)$ [39]

In the above example of optimization, the maximum number of allowed function evaluations was set to 25000 i.e. 1250 iterations, desired accuracy was 100%, neighborhood acceleration (c3) set to 1, constriction factor was set to 0.729; the inertia weight ω was gradually decreased from 1 towards 0.1 with c1 = c2 = 2, V_{max} = 4 and the size of the Swarm were selected to be 20. The range of variables was bounded between [-100 to 100]. The aforementioned values for all PSO parameters are considered default values, and they are used widely in the relevant literature [11].

The results in Figure 3.2 indicate that using the Integer PSO for optimization of Integer problems can give solutions to Integer based problems. As can be seen Integer PSO takes 205 iterations to reach the global minimum of -737.the number of iterations taken by integer PSO is indeed high. Since this is a very simple example, the amount of time is

really a fraction. But for problems that take 30-40 minutes for each iteration, 200 iterations would mean just inefficient spending of resources.

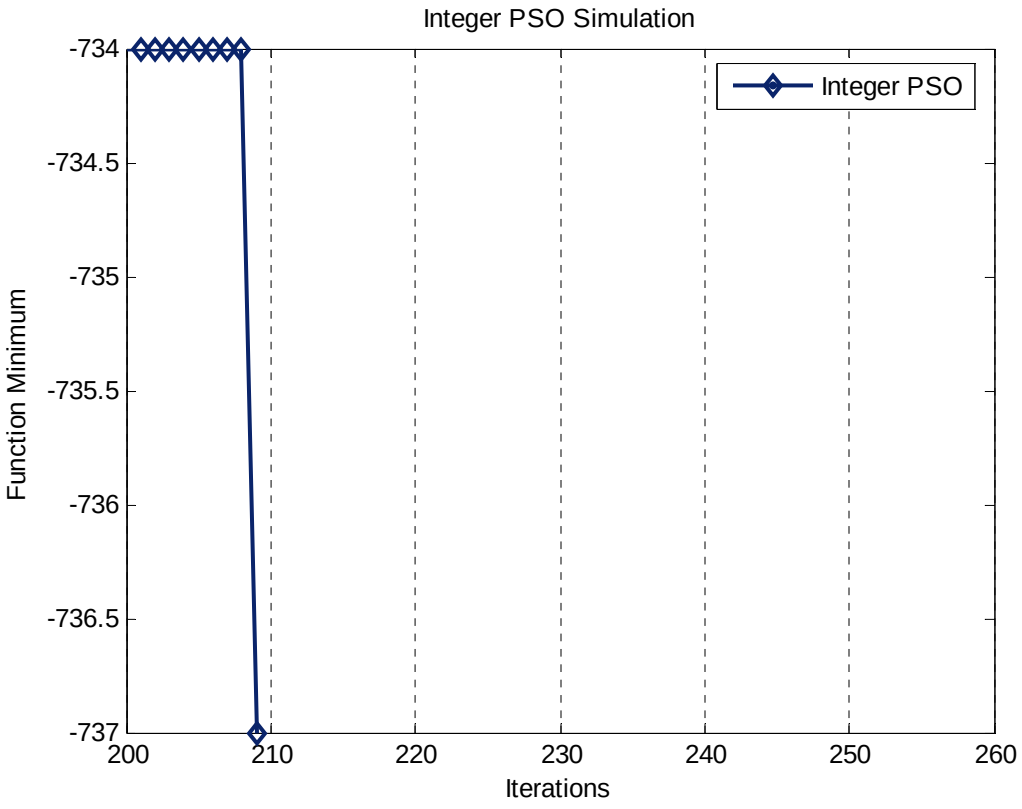

Figure 3.2 Integer PSO Example

Thus the integer PSO cannot be relied upon alone, we have o build up PSO which can guarantee us faster convergence and in very small number of iterations.

3.2 Hybrid-Integer PSO

In the previous section it has been shown that Hybrid PSO is superior to Basic PSO with all modifications. Since the problem surrounding our Process (RDC) involves variables which are pure integers, it is of prime importance that the PSO algorithm should be able to converge onto a satisfactory integer value for any problem which is discrete in nature. As a test to the Hybrid-PSO, we have chosen a fitness function cited in the literature [29] to check whether hybrid PSO can converge to optimal integer solution.

- As an example, $F(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$ with range for x as $x \in [-100,100]$ and solution $F(x) = -6$ at $x = (2,-1)$

As can be seen from the Figure 3.3 both the standard PSO and the Hybrid PSO fail to converge to the required Global minimum. Thus it is necessary to amalgamate Integer PSO as discussed in section 3.1.1 with Hybrid PSO 3.1.2. Thus we came up with our Algorithm termed HI-PSO (Hybrid-Integer PSO).

In HI-PSO the initial Swarm size is incremented to large size compared to its actual size and the Integer PSO is operated upon this Swarm. After the evaluation of the fitness function the best swarms are selected out of the total flock of swarms that was created in the beginning. The criterion of selection is the fitness achieved for the particular function. After the selection rest of Swarm is neglected. This is like ‘Selection or ‘Survival of the

fittest'. The main advantage of incrementing the Swarm to a very high initial number is that the Swarms cover an imminent large area in the beginning and based upon the fitness achieved the selected Swarms move in the direction of the optimal solution.

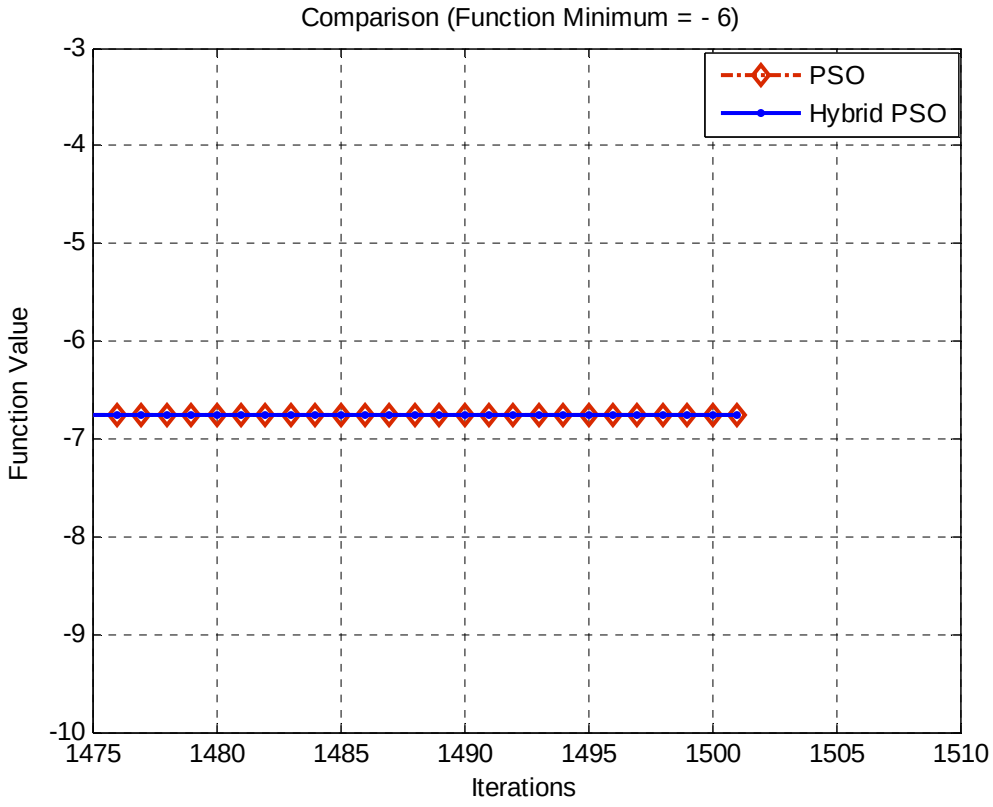

Figure : 3.3 Comparison of PSO with Hybrid PSO

Thereafter the positions and the velocities of the selected Swarm are updated according to the Basic - PSO operators. Note that the size of swarm is now the actual size after stripping away the Un-fit particles from the selection. These new particles are breaded and a sub-population among them is formed based on the algorithm discussed in the previous section of Hybrid PSO. The new particles formed after the Hybridization are

rounded to the nearest integers and then they are evaluated by the function and the fitness or termination criterion is tested. If it is satisfied the algorithm stops other wise the algorithm repeats going back to hybridization-rounding step and re-evaluated until the termination criterion is met.

- As a test we recall the previous example $F(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$ with range for x as $x \in [-100,100]$ and solution $F(x) = -6$ at $x = (2,-1)$

With the same parameters of PSO as tested before the New HI-PSO algorithm is tested. The results can be seen in the Figure 3.5. In comparison we have selected Basic PSO which does not converge even at the end of total iterations to the global minimum and the Integer PSO which converges to the Global Minimum but takes around 15 iterations whereas the HI-PSO takes only 4 iterations to converge to the global minimum.

The success is mainly attributed to the Increment in the size of the Swarm which is done initially and partially to the Hybridized model. The main reason for this bias is that at the beginning the Swarm tries to span the maximum possible search area and incrementing the Swarm size initially to large number increases the probability of searching the span with a good hit. The hybridized model however increases the convergence rate near the end and avoids stagnation in the local optima

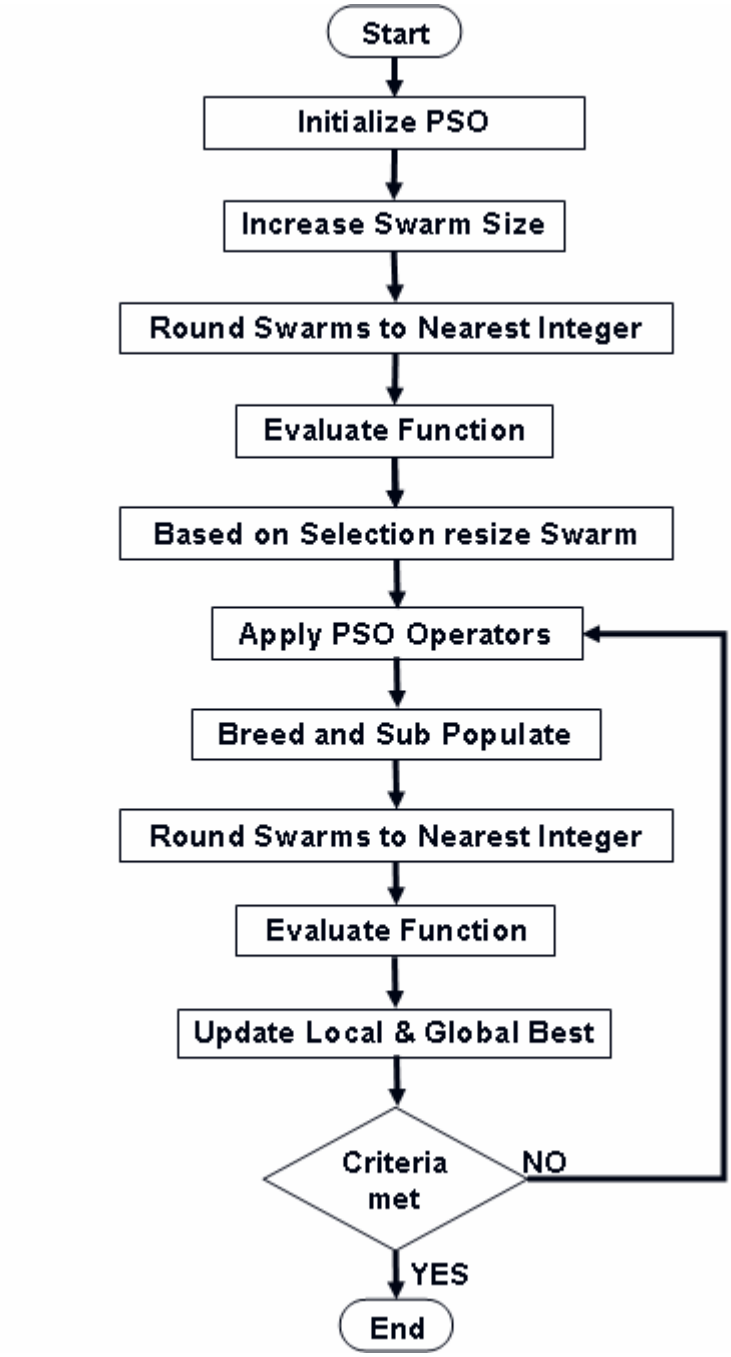

Figure : 3.4 Algorithm of Hybrid Integer PSO

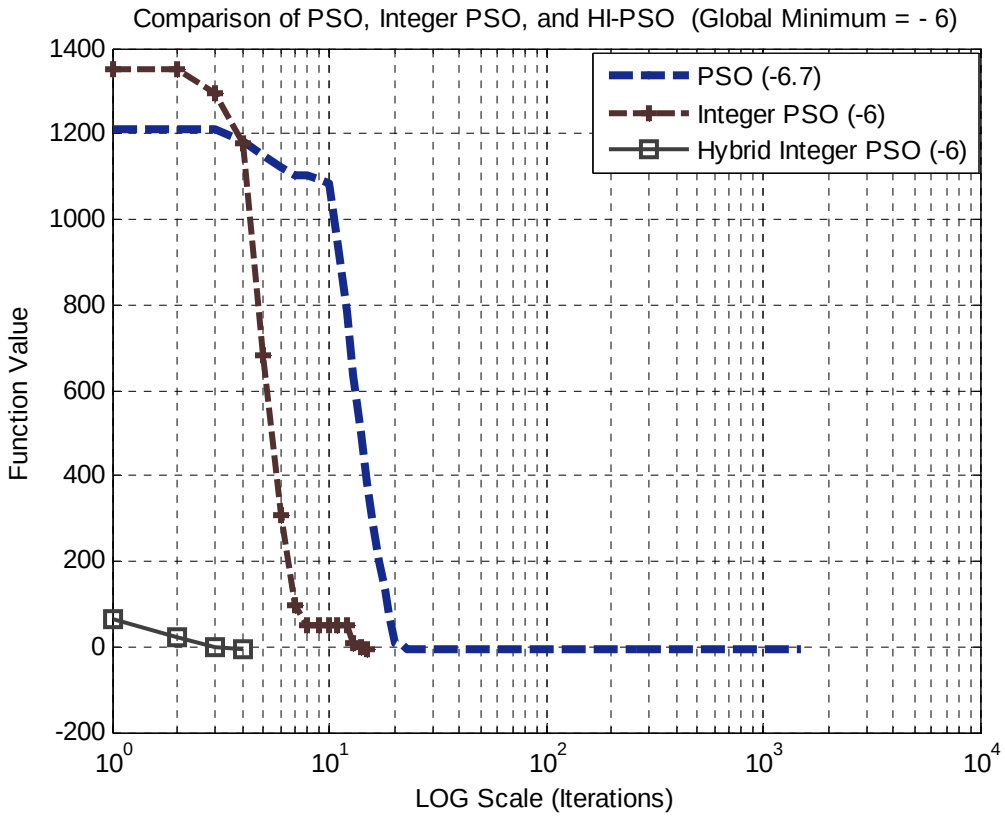

Figure 3.5 Hybrid Integer PSO vs basic PSO and Integer PSO

3.3 Test Functions

Let us now test the HI-PSO algorithm for standard test functions as cited in the literature and compare them with the results of standard literature to which we have referred. The following are the five standard test problems that were used from the literature to evaluate the working of our HI-PSO algorithm on Integer programming

- $F_1(x) = x^T x$ the range of x is $x \in [-100,100]$ and global solution is $F_1(x) = 0$ with $x = (0...0)$ [40]
- $F_2(x) = -(15\ 27\ 36\ 18\ 12)x + x^T \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix}$

The range of x is $x \in [-100,100]$ and the solutions is $F_2(x) = -737$ with $x = (0,11,22,16,6)\ and\ x = (0,12,23,17,6)$ [39]

- $F_3(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2$ the range of x is $x \in [-100,100]$ with solution $F_3(x) = 0$ at $x = (1,1)$ [39]
- $F_4(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$ with range for x as $x \in [-100,100]$ and solution $F_4(x) = -6$ at $x = (2,-1)$ [29]

$F_5(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$ where x has a range given by $x \in [-100,100]$ with solution $F_5(x) = -3832.68$ at $x = (0,1)$ [39]

Table 3.1 Laskari et al. PSO parameters

Function	Dimension	Swarm Size
F_1	5	10
F_2	5	70
F_3	2	20
F_4	2	10
F_5	2	20

Max. Function Evaluations = 25000, Constriction Factor = 0.729, Inertia Weight: 1 to 0.1 $C_1=C_2=2$, $V_{max} = 4$. The above settings were applied by the author in his work, we have retained the settings and added the Hybrid settings of initial Swarm Size Increment = Initial + 100 Swarms and Probability of Breeding = 0.25.

Table 3.2 Performance of Hybrid Integer PSO

Function	Method	Success	Mean	St.D.	Dim	Swarm
F1	PSO-HI	30/30	279.6	67.33	5	10
	PSO-Lit	30/30	418.3	83.9	5	10
F2	PSO- HI	30/30	3084	215.59	5	70
	PSO-Lit	30/30	3171	493.6	5	70
F3	PSO- HI	30/30	182.66	77.67	2	20
	PSO-Lit	30/30	302	80.5	2	20
F4	PSO- HI	30/30	174.6	92.91	2	10
	PSO-Lit	30/30	191	65.9	2	10
F5	PSO- HI	30/30	224.6	113.2	2	20
	PSO-Lit	30/30	306.6	96.7	2	20

The results in the Table 3.2 were taken from 30 experiments and success rate of 100% was achieved similar to the author. Mean and standard deviations of function evaluations can be seen in the table. The results are better compared to the work done by Laskari et al. The functions 1 and 2 are 5 dimension problems i.e. five variables problems and the

remaining are 2 dimensional problems. Our algorithm stands good to all of these functions.Hence it is conclusive that an initial large Swarm followed by a breeding it can do the job better provided it is properly tuned for integer problems.

CHAPTER 4

OPTIMIZATION PROBLEM FORMULATION

In industrial plant operations, benefits arise from improved plant performance such as improved yields of valuable products (or reduced yields of contaminants), reduced energy consumption, higher processing rates, and longer times between shutdowns. Optimization can also lead to reduced maintenance costs, less equipment wear, and better utilization of the process overall. It is extremely helpful to systematically identify the objective, constraints and degrees of freedom in a process or a plant, leading to such benefits as improved quality of design, faster and more reliable trouble shooting, and faster decision making [1].

Predicting benefits must be done with care. Design and operating variables in most industrial plants are always coupled in some way or another. If the fuel bill for generator or a distillation column is $3000 per day, a 5 percent savings may justify an energy conservation project. However, if the process is operated in units then it is incorrect to just sum up the factors that consume the input and claim a percentage reduction in total [1].

Among the various hierarchies of optimization the design engineers are one. Engineers engaged in the process design and equipment specification are concerned with the choice of a process and a nominal operating condition. They are responsible for basic design, amount of input required, the configuration of the system and assembly of the system so as to get the operating efficiency to maximum. The simulation tools play a handy role in helping these engineers achieve the objectives. The other hierarchies may include the management and plant operations [1].

A significant example of industrial process is Reactive distillation column. The next few sections will entail about the physical structure process description mathematical modeling and chemical view point. Section 4.1 is in general about the physical setup of reactive distillation column. This section also contains the mathematical modeling the state space design and the optimum design data. Section 4.2 details about the cost function formulation.

4.1 Reactive Distillation Column

Reactive Distillation is the process of separating reactants, to give products in the common chamber. It is a non-Linear and complex process to design. The Chemical industry has already acknowledged its significance due to high gains and its compact nature [1]. Pre-Installation optimal design of this process is of great concern because it is

an installation of one time, but it requires constant supply of materials like fuel and reactants, out of which fuel is very costly. A saving in the design of an ideal reactive distillation column (Ideal RDC) [20] without compromising any of the desired features would indeed be a great profit to the industry. In the chemical process industries, chemical reaction and purification of the desired products of distillation are usually carried out sequentially. In many cases, the performance of this classic chemical process can be significantly improved by integration of reaction and distillation in a single multifunctional process unit. This integration concept is called 'reactive distillation' [2]. Increased process efficiency and reduction of investment and operational costs are the direct results of this approach. Some of these advantages are realized by using reaction to improve separation; others are realized by using separation to improve reaction.

RDC is an ideal two-reactant-two-product reactive distillation column proposed by Al-Arfaj and Luyben [3] and later developed into state space model by Olanrewaju [41]. It consists of a reactive section in the middle and non-reactive rectifying and stripping sections at the top and bottom. The column consists of Reactive Trays (N_{RX}) in the middle, Rectifying Trays (N_R) in the Top and Stripping Trays (N_S) in the bottom. The column is numbered from reboiler to condenser. The reaction that takes place in the reactive zone is exothermic liquid-vapor in nature and is given by

$$A + B \Leftrightarrow C + D \tag{4.1}$$

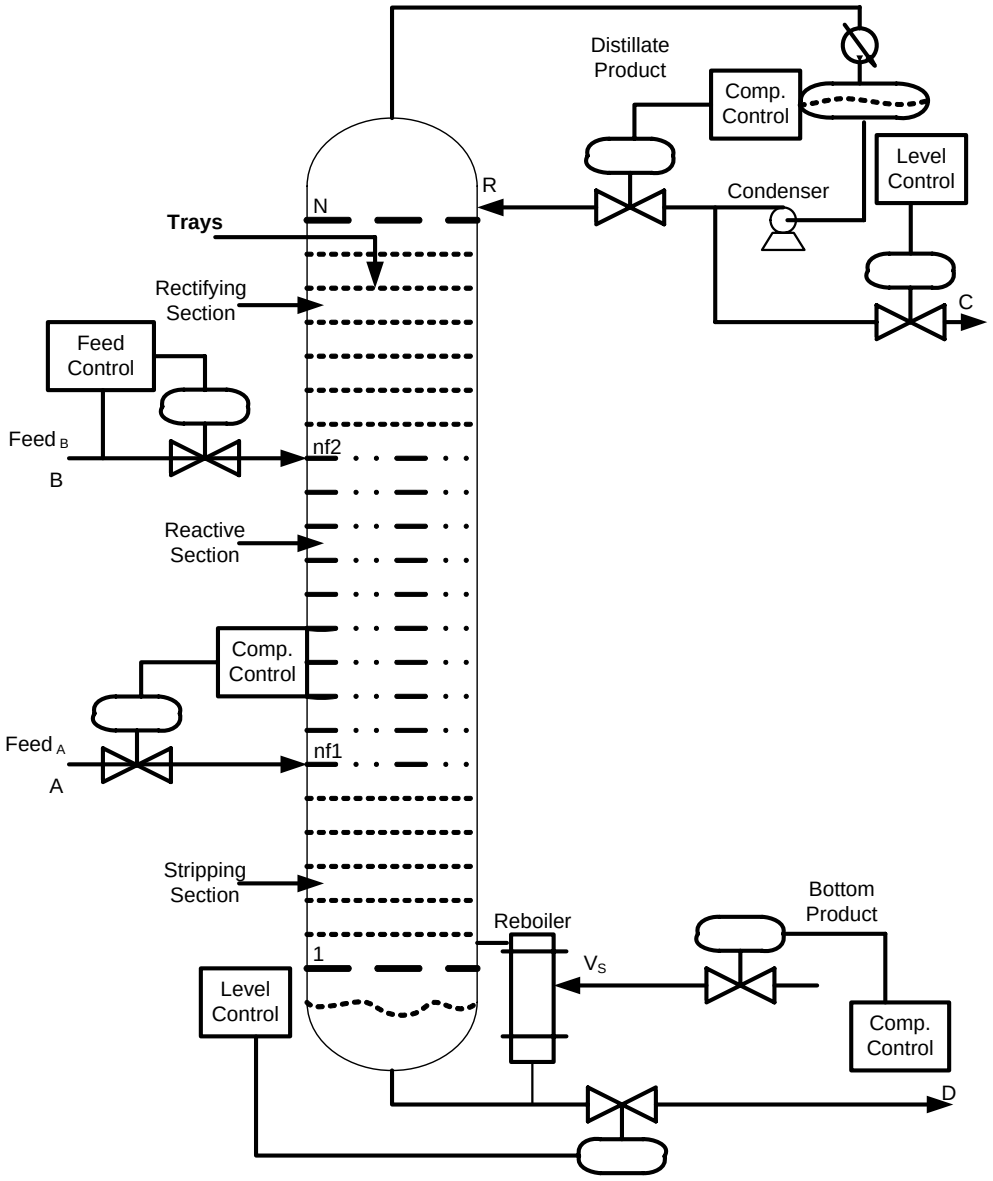

Figure 4.1 Reactive Distillation Column

During the process of distillation, the reactant B which is fed through one of the input feeds is recovered in the rectifying section from the output product C. In the stripping section the reactant A which is the second feed is recovered from output product D, in

the reactive section which comprises the middle section of the reactive distillation column the products are separated and any undesired reaction between reactants A and B and products C and D is prevented.

The volatilities of the products and reactants are such that

$$\alpha_C > \alpha_A > \alpha_B > \alpha_D \tag{4.2}$$

Where α is the volatility of the Component as can be observed from the above relation C is the lightest product with highest volatility and D is the heaviest with lightest volatility along with A and B in between them. This relative volatility ensures that the products A and B have high concentration in the reactive section, which is typical example of an Ideal Reactive Distillation column.

The three sections consist of trays that have different Composition Profiles, Vapor Profile, Liquid Profile and Hold Up. Along the left side two inputs can be seen marked as $Feed_A$ and $Feed_B$. these are the two input streams which feed the reactants A and B, the reactant A is Lighter compared to B, so when the feeds enter the column, A tries to go up and since B is heavier than A it tries to settle in the bottom part of the column.

The controllers in the process are termed as dual end composition control structure. The reflux drum level is controlled by manipulating the distillate flowrate. The purity of both products is maintained at 95%. In the distillate products, the composition of component C

in the distillate is controlled by manipulating the reflux flowrate from the condenser, while the bottoms composition of component D is controlled by manipulating the vapor boilup.

4.1.1 Process Modeling

The dynamic model of a Reactive Distillation Column consists of a large number of Non-Linear Differential equations and demands information about the system such as composition of trays, vapor, liquid flow rates, liquid holdup in all stages at every instant, tray hydraulics, energy balances, and vapor liquid equilibrium data. However the system that is being investigated is an ideal generic reactive distillation with simple vapor-liquid equilibrium, reaction kinetics and physical properties [41]. The model assumptions are as follows

1. Ideal Vapor-Liquid Equilibrium
2. Saturated Liquid Feed and Reflux flowrate
3. The energy equations are neglected by assuming constant molar overflow except in the reactive zone where the vapor flow rate increases because of the heat of reaction which vaporizes some liquid on each tray.
4. Constant relative volatilities.
5. Fixed heat of reaction and vaporization and saturated liquid feed and reflux.

The reactive distillation model is based on dynamic mass balance. While the energy equations are neglected by assuming constant molar overflow except in the reactive zone. Therefore the nonlinear state space model can be described as follows

1. Reboiler:

$$\frac{d\left(M_B x_{Bj}\right)}{dt} = L_1 x_{1j} - B x_{Bj} - V_m y_{Bj} \qquad \text{j= 1,2,…, Nc-1} \tag{4.3}$$

$$\frac{dM_B}{dt} = L_1 - V_m - B \tag{4.4}$$

Where M is liquid holdup, x_{ij} is the liquid mole fraction of component j on tray i, t is the time in sec, L is the liquid flow rate, B is the bottom flow rate, V is the vapor flow rate, i is the tray number, j is the component number, $_B$ is the bottom tray and Nc is total number of components which is equal to four.

2. Stripping section, tray i (1< i < Ns+1)

$$\frac{d\left(M_i x_{ij}\right)}{dt} = L_{i+1} x_{i+1,j} - L_i x_{i,j} + V_m y_{i-1,j} - V_m y_{i,j} \qquad \text{j= 1, 2,…, Nc-1} \tag{4.5}$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i \tag{4.6}$$

Where N_S is the number of trays in stripping section.

3. Reaction section

Feed Tray for reactant A :(i = N_{S+}+1)

$$\frac{d\left(M_i x_{ij}\right)}{dt} = L_{i+1}x_{i+1,j} - L_i x_{i,j} + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j} + F_{oA}Z_{oA\,j} \quad \text{j= 1, 2,…, Nc-1} \qquad (4.7)$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda / \Delta H_v + F_{oA} \qquad (4.8)$$

Where y_{ij} is the vapor mole fraction of component *j* on tray *I, R* is the reflux rate, F_{oA} is the feed flow rate of A and Z_{oA} is the Feed composition of A, λ is heat of reaction and ΔH_v is the heat of reaction

Tray i :(Ns+2< i < N_S+1+N_{RX})

$$\frac{d\left(M_i x_{ij}\right)}{dt} = L_{i+1}x_{i+1,j} - L_i x_{i,j} + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j} \quad \text{j= 1, 2,…, Nc-1} \qquad (4.9)$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda / \Delta H_v \qquad (4.10)$$

Feed Tray for reactant B (i = N_{S+}+1+N_{RX})

$$\frac{d\left(M_i x_{ij}\right)}{dt} = L_{i+1}x_{i+1,j} - L_i x_{i,j} + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j} + F_{oB}Z_{oB\,j} \quad \text{j= 1, 2,…, Nc-1} \qquad (4.11)$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda / \Delta H_v + F_{oB} \qquad (4.12)$$

N_{RX} is the total number of reactive trays in middle section.

4. Rectifying section, tray i :

($N_S+2+N_{RX} < i < N_S+1+N_{RX}+N_R$)

$$\frac{d\left(M_i x_{ij}\right)}{dt} = L_{i+1}x_{i+1,j} - L_i x_{i,j} + V_n y_{i-1,j} - V_n y_{i,j} \tag{4.13}$$

j= 1, 2,…, Nc-1

$$\frac{dM_i}{dt} = L_{i+1} - L_i \tag{4.14}$$

$$V_i = V_{i-1} - R_{i,j}\lambda / \Delta H_v \tag{4.15}$$

Where N_R is the total number of rectifying trays in the top section.

5. Condenser and Reflux drum:

$$\frac{d\left(M_{D,j} x_{D,j}\right)}{dt} = V_n y_{Nr} - (R+D)x_{D,j} \quad \text{j= 1, 2,…, Nc-1} \tag{4.16}$$

$$\frac{dM_D}{dt} = V_n - R - D \tag{4.17}$$

Where D is he distillate flow rate.

Liquid flowrate is calculated from a linearized form of the Francis Weir formula:

$$L_i = \overline{L}_i + \frac{M_i - \overline{M}_i}{\beta} \tag{4.18}$$

Where β as hydraulic time constant.

Net reaction rate of component j on tray i is given as:

$$R_{i,j} = M_i \left(k_{F,i} x_{A,i} x_{B,i} - k_{B,i} x_{C,i} x_{D,i} \right) \tag{4.19}$$

The forward and backward specific reaction rates (kmol.s^{-1}.kmol^{-1}) on tray i:

$$k_{F,i} = a_F e^{-E_F / RT_i} \tag{4.20}$$

$$k_{B,i} = a_B e^{-E_B / RT_i} \tag{4.21}$$

Where a_f and a_B are the pre-exponential factors, E_f and E_B are the activation energies, and T_i is the absolute temperature on tray i.

Liquid-vapor equilibrium equation:

$$y_{i,j} = \alpha_j x_{ij} / \sum_{k=1}^{Nc} \alpha_k x_{ik} \quad \text{j= 1, 2,…, Nc-1} \tag{4.22}$$

mole fraction:

$$x_{i,Nc} = 1 - \sum_{k=1}^{Nc-1} x_{ik} \tag{4.23}$$

Temperature on tray i:

$$T_i = B_{vp,j} / [A_{vp,j} - \ln(\alpha_j P / \sum_{k=1}^{Nc} \alpha_k x_{ik}) - B_{v,j}] \quad (4.24)$$

where A_v and B_v are Antoine constants .Equation 24 is obtained by assuming Raoult's law and a vapor pressure of pure component j of the following form:

$$\ln P^0{}_j = A_{vp,j} - \frac{B_{vp,j}}{T} \quad (4.25)$$

4.1.2 State space Model

The equations which describe the process model as presented in section 4.1.1 can be made more compact and rearranged into state space model form. This is done by substituting the total material balances (Eq. 4.4, 4.6, 4.8, 4.11, 4.13, 4.15, 4.17, and 4.19) into the component balances to give the equations:

$$\frac{dx_{Bj}}{dt} = \left[L_1(x_{1j} - x_{Bj}) + V_m(x_{Bj} - y_{Bj})\right] / M_B \qquad \text{j= 1, 2,…, Nc-1, i = B} \quad (4.26)$$

$$\frac{dx_{ij}}{dt} = \left[L_{i+1}(x_{i+1,j} - x_{i,j}) + V_m(y_{i-1,j} - y_{i,j})\right] / M_i \qquad \text{j= 1, 2,…, Nc-1, } (1 \leq \text{i} < \text{Ns+1}) \quad (4.27)$$

$$\frac{dx_{ij}}{dt} = \left[L_{i+1}(x_{i+1,j} - x_{i,j}) + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j} + F_{oA}Z_{oA}\right] / M_i \quad (4.28)$$

j= 1, 2,…, Nc-1,i=N_S+1

$$\frac{dx_{ij}}{dt} = \left[L_{i+1}(x_{i+1,j} - x_{i,j}) + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j}\right]/M_i \quad \text{j= 1, 2,…,Nc-1,} \tag{4.29}$$

$$(\text{Ns+2} \leq \text{i} < \text{N}_\text{S}\text{+1+N}_\text{RX})$$

$$\frac{dx_{ij}}{dt} = \left[L_{i+1}x_{i+1,j} - L_i x_{i,j} + V_{i-1}y_{i-1,j} - V_i y_{i,j} + R_{i,j} + F_{oB}Z_{oB\,j}\right]/M_i \quad \text{j=1,2,…,Nc-1,} \tag{4.30}$$

$$(\text{i=N}_{\text{S+}}\text{+1+N}_\text{RX)}$$

$$\frac{dx_{ij}}{dt} = \left[L_{i+1}(x_{i+1,j} - x_{i,j}) + V_n(y_{i-1,j} - y_{i,j})\right]/M_i \qquad \text{j= 1, 2,…, Nc-1} \tag{4.31}$$

$$(\text{Ns+2+N}_\text{RX} \leq \text{i} \leq \text{Ns+1+N}_\text{RX}\text{+N}_\text{R})$$

$$\text{Nr=Ns+1+N}_\text{RX}\text{+N}_\text{r} \tag{4.32}$$

$$\frac{dx_{D,j}}{dt} = \left[V_n(y_{Nr} - x_{D,j})\right]/M_D \tag{4.33}$$

$$\frac{dM_B}{dt} = L_1 + V_m - B \tag{4.34}$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i \tag{4.35}$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda/\Delta H_v + F_{oA} \quad \text{i} = \text{N}_{\text{S+}}\text{+1} \tag{4.36}$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda/\Delta H_v \tag{4.37}$$

$$\frac{dM_i}{dt} = L_{i+1} - L_i + R_{i,j}\lambda/\Delta H_v + F_{oB} \quad \text{i} = \text{N}_{\text{S+}}\text{+1+N}_\text{RX} \tag{4.38}$$

$$\frac{dM_D}{dt} = V_n - R - D \tag{4.39}$$

$$Y=[x_{Dj,}x_{Bj,}]^{1} \quad (4.40)$$

Thus, nonlinear state space models would be of the form:

$$\frac{dX(t)}{dt} = f(X(t),U(t),d(t);\theta) \quad (4.41)$$

$$Y(t)=h(X(t)) \quad (4.42)$$

Where f and h are nonlinear functions of process model.

4.1.3 Optimum Steady State Design

An optimum steady state design for one column reactive distillation process for our work will be considered based on the design of Al-Arfaj and Luyben [3] and developments made by Olanrewaju [41]. The purities of both the product streams will be assumed to be 95%. Reactive tray holdup will be assumed to be 1 kmol for fresh feed flow rates of 12.6 mol/s of both A and B. Table 4.1 gives the kinetic, physical property, and vapor-liquid equilibrium parameter values. Table 4.2 gives the detailed optimum design for single-column process.

Table 4.1. Physical Properties

Activation energy (cal\mol)	
forward	30 000
backward	40 000
Specific reaction rate at 366k ($Kmols^{-1}Kmol^{-1}$)	
forward	0.008
backward	0.004
heat of reaction (cal/mol)	-10 000
heat of vaporization (cal/mol)	6944
relative volatilities	
α_A	8
α_B	4
α_C	2
α_D	1

vapor pressure constants	C	A	B	D
Avp	13.04	12.34	11.45	10.96
Bvp	3862	3862	3862	3862

Table 4.2. Optimum Design for single-column process

Flow rate (mol/s)				
vapor boil up (mol/s)	28.14			
reflux (mol/s)	32.78			
distillate (mol/s)	12.6			
bottoms (mol/s)	12.6			
pressure (bar)	9			
tray holdup (Kmol/s)	1.00			
Tray numbers				
Stripping	7			
Reactive	6			
Rectifying	7			
	A	B	C	D
Composition (mole fraction)				
distillate	0.04755	0.00246	0.94999	0.00000
bottoms	0.00164	0.04839	0.00000	0.94999

4.2 Cost Function Formulation

The RDC requires a steady temperature inside the column for separation to take place. This separation is obtained by the method of distillation. The high temperature inside the column is maintained by the use of reboiler which requires steady supply of fuel. The amount of fuel consumed per hour to keep the reboiler burning that can maintain a steady temperature inside the column and produce products of desired purity constitutes the optimization problem. The ‘cost of fuel consumption’ as an optimization variable is more significant compared to the ‘cost of pre-installation hardware design’, reason being, the

process is one time installation, installation cost compared to the expenditure on fuel over a long time is small. Hence an optimization of hardware design for low fuel consumption is very much necessary and significant.

The task of the rectifying section (Top) is to recover reactant B from the product stream C. In the stripping section (Bottom), the reactant A is stripped from the product stream D. In the reactive section the products are separated, driving the equilibrium and preventing any undesired side reactions between the reactants A (or B) with the product C (or D). Therefore, reactants A and B are intermediate boilers while product C is the lightest and product D is the heaviest. This ensures that high concentration of the reactants A and B is maintained in the reactive zone, which is typical for reactive distillation application. The reactive section contains N_{RX} trays. The rectifying section contains N_R trays, and the stripping section below the reactive section contains N_S trays. The column is numbered from the reboiler to the condenser.

A dual composition control suggested by Al-Arfaj and Luyben [20] is implemented to obtain the desired manipulated variables. Composition of product C in the distillate is controlled by manipulating the reflux flowrate, while the vapor boilup is manipulated to control the bottoms composition of component D. The controllers automatically manipulated both the reflux flowrate and vapor boilup to the values that correspond to the desired conversion and purity.

The vapor boilup (V_s) in a reactive distillation column is the amount of vapor flowing from the reboiler into the distillation column. The amount of heat required to vaporize depends upon the design parameters of the reactive distillation column like number of trays in each of the three sections, the feed locations and the pressure inside the column. Hence reducing vapor flow or vapor boilup (V_s) is reducing the amount of fuel consumed. Nevertheless the quality of components cannot be sacrificed at any price. Hence a stringent concentration measure of 95% is desired from the system. The vapor boilup is used to control the bottom composition and the reflux rate is used to control the Distillate composition. The reflux rate is not optimized. The reason is that the reflux drum is used for cooling to condense the impurities which are present in vapor form and send them back to the reactive distillation column for re-processing and re-purification. The cooling is done by using water jackets and the cost of raw water is negligible compared to the cost of fuel per day, hence the reflux rate can be neglected.

The fuel consumed by the RDC is equal to the amount of fuel required by the reboiler to vaporize the liquid and send it back in to the Distillation column.

$$V_S = vss(1) * 2 * C_B \tag{4.43}$$

Where $vss(1)$ is the vapor holdup of the tray-1(Bottom) at steady state and C_B is the Boiler Constant which is used to control the amount of vapor supplied by the reboiler so

as to obtain the desired concentration. The factor C_B which is obtained from the reboiler controller equation checks the concentration of the products and adjusts the reboiler duty.

$$C_B = 0.5 + K_B \bullet Er_B + Eri_B \tag{4.44}$$

$$K_B = 1.54 \tag{4.45}$$

$$Er_B = \frac{(0.95 - Xblag2)}{0.2} \tag{4.46}$$

Given are $Xblag2$ Internal Controller Lag and Eri_B Internal Controller Error. The steady state vapor boilup is obtained after the Non-Linear equation describing the process is solved (Eq 4.1 to 4.25). which are dependent up on N_S (Stripping Trays), N_{RX} (Reactive Trays), N_R (Rectifying Trays), P (Pressure), N_{F1} (Feed Location for Input A) and N_{F2} (Feed Location for Input B). *vss* is calculated by equation 4.15 recursively.

$V_i = V_{i-1} - R_{i,j}\lambda / \Delta H_v$, Where $R_{i,j} = M_i\left(k_{F,i}x_{A,i}x_{B,i} - k_{B,i}x_{C,i}x_{D,i}\right)$

The remainder terms are calculated recursively from the non-Linear Differential equations from Eq. 4.1 to 4.25

The objective functions can be defined as

I. $V_S = f(N_S, N_{RX}, N_R)$ for objective 1

II. $V_S = f(N_S, N_{RX}, N_R, P)$ for objective 2

III. $V_S = f(N_S, N_{RX}, N_R, P, N_{F1}, N_{F2})$ for objective 3

Subject to the condition that concentration of the products is maintained at 95%.

CHAPTER 5

SIMULATION AND SETUP

This chapter contains the simulation setup and the results along with discussions.

Section 5.1 gives a brief description about the lab environment for simulation. Section 5.2 describes the optimization setup procedure, this section details about the parameters, the algorithm, reactive distillation column setup, initial values criterion, range of variables used and the process run time. Section 5.3 enlists the objective functions section 5.4 illustrates an algorithm that is used to solve the problem. Finally section 5.5 displays he results and carries the discussions.

5.1 PC Setup

The simulations were carried on over Matlab 6.5 on IBM Compatible PC's of 512 MB RAM and Pentium 4 Processors with 2 Giga hertz speed. While the simulation were being run entire processor was dedicated to Matlab, i.e. 100% Processor time was dedicated to matlab alone in order to have the maximum efficiency out of the computers.

A cluster of 18 IBM PC's of identical configurations and a same version of Matlab was used to check the accuracy of simulations. The cluster was handy when evaluation of

objectives was carried by brute force (Exhaustive search) method, thus reducing the execution time by a factor of 18.

5.2 Optimization Setup

5.2.1 HI-PSO

The variables of the Process were pure integers; HI-PSO was applied to optimize the process. The algorithm was used as was mentioned in the previous chapter with these settings.The Hybrid-Integer Particle Swarm Optimization with V_{max} operator, Inertia Weights, Constriction Factor, Neighborhood and Integer Programming along with breeding and sub-population was used. The simulation was run for a length of 100 iterations and 10 such experiments were carried in order to check the accuracy of the results. The termination criterion was completion of total iterations i.e. 100.

- Swarm = 20;
- Iterations = 100;
- Cognitive Acceleration (c1) = 2;
- Social Acceleration (c2) = 2;
- V_{max} = 4;
- Inertia weights = 1 to 0.1
- Constriction Factor: 0.729
- Neighborhood: Circular
- Initial Swarm: 120
- Breeding Probability : 0.2
- Number of Variables:

- Objective 1 = 3 (Integer)
- Objective 2 = 4 (3 Integer + 1 Real)
- Objective 3 = 6 (5 Integer + 1 Real)

5.2.2 Reactive Distillation Column

The Model Proposed by Al-Arfaj and Luyben [20] and Olanrewaju [41] has been used. Retaining all its originality the model was made variable dependent to accommodate the change in number of trays (N_S ,$N_{RX,}$ N_R) and Pressure(P) and the Feed Locations(N_{F1} N_{F2}). The parameters are given in the chapter 4.

5.2.2.1 Initial Values

As was mentioned earlier, the process is Non-Linear in nature and the order of the process changes with the change in trays in any of the section. Hence this forms a new model everytime any of the 3 sections tray quantity is varied. Thus the process requires that initial values of Liquid profile, composition profile vapor profile and holdup per tray be provided. This necessity that at every new model there should be initial values present is fulfilled by extrapolating the values from the base case.

The base case had Ns = 7, Nrx = 6 and Nr = 7, so the initial values of

1. Component Composition (22x4)
2. Holdup (22x1)
3. Vapor (22x1)
4. Liquid (22x1)

Profiles consisted of 20 + 2 rows (values), the two extra being the bottom bed and the top collector which also serve as tray. Ns becomes Ns+1(due to bottom bed) and Nr becomes Nr+1(due to collector). Ns now becomes 8 Nrx remains same to be 6 and Nr becomes 8.

To find Initial Values of Ns for a Given Model:

1. Find the average of consecutive locations in the Base Ns+1 Values.
2. Insert them in between the two actual locations.
3. Repeat till you reach the end of Ns+1, this gives a total of 2Ns+1 Values,
4. Repeat the above procedure from first step till the length of Ns becomes greater than or equal to 25

As an Illustration, the 8 values of Ns+1 will generate 7 values after averaging and after re-insertion in between the source locations the total length becomes 15. upon repeating the above procedure we will have 15+14 values in Ns. So that becomes 29 which is greater than 25. The reason we are stopping at greater than 25 is that the maximum range of any section is between 5 and 25 trays.

5. From the stack of these 29 numbers, the selection is made as required based upon the Model order.
6. For example: in order to have Ns = 10, calculate the Step = 29/10 = 2.9.
7. From the stack of 29 Ns, take step of 2.9 until 29 is reached.
8. since 2.9 is not a perfect integer, take the average of the floor and ceiling around the number 2.9
9. so the first number will be average of 2nd and 3rd value from the Stack of 29
10. Repeat this loop until the desired vector of initial values is obtained.

The same principle applies to the Nrx and Nr part. From the above procedure all the four initial value sets can be calculated.

5.2.2.2 Range of Variables

The range of variables used in the simulations is given below:

- N_S - Stripping Section [5 to 25]
- N_{RX} - Reactive Section [5 to 25]
- N_R - Rectifying Section [5 to 25]
- P – Column Pressure [5 to 15]
- N_{F1} - Stripping Section [N_S - 5 to $\frac{N_{RX}}{2}$]
- N_{F2} - Stripping Section [$\frac{N_{RX}}{2}+1$ to $N_S + N_{RX} + 5$]

For all the three objectives this range was maintained.

The feed location is dependent upon the selected Ns and Nrx. Hence a relation describes the range of feeds rather than a fixed integer. In the optimization Algorithm for the third objective the feeds are selected after the selection of Ns, Nrx, Nr and P according to the relation mentioned above.

5.2.2.3 Process Run Time

The process run time is the simulation of process to produce the desired products for given hours. It is not the time for which the optimization algorithm is run. Any change in optimization Variables will cause change in the order of the system and the Non-Linear equations describing it. In order to use such a model an initial estimate of the composition profile is necessary. We have developed a method which extrapolates the initial values based on the RDC system taken from the Literature. This estimate serves as a disturbance in the beginning to the system; the internal composition controllers control this error and bring the system back to steady state.

The amount of time required by the system to come back to steady state depends upon the order of the system hence a considerable time lapse should be provided so that any system within the specified range can come back to steady state after the initial values are used. Since we did not use any criteria to determine the relation between the system and

the time required we had to let the system run for a long enough time so that any design based on trays can come to steady state.

Not all the combinations from the specified range can give the desired products purity, sufficient time has to be given to check for steady state stability for the possible configurations that can give desired purities. The system was initially simulated with three sets, 8, 16 and 32 hours of run time.

8 Hours: When the Process Run Time was set to 8 Hours and Brute force method was applied to check all the possible combinations the best systems did not reach the steady state. This can be seen in the Figure 5.1. Almost all the combinations are still oscillatory at the end of 8 hours although the error is very less to get an exact accuracy we must always apply stringent measures. Simulation time for each combination with process run time of 8 hours was found to be 40 seconds.

16 Hours: Further proceeding with the investigation Process Run Time was doubled to 16 hours and was observed that almost all are combinations are around accuracy of 10^{-6}.

Simulation time for each combination with process run time of 16 hours was found to be 70 seconds.

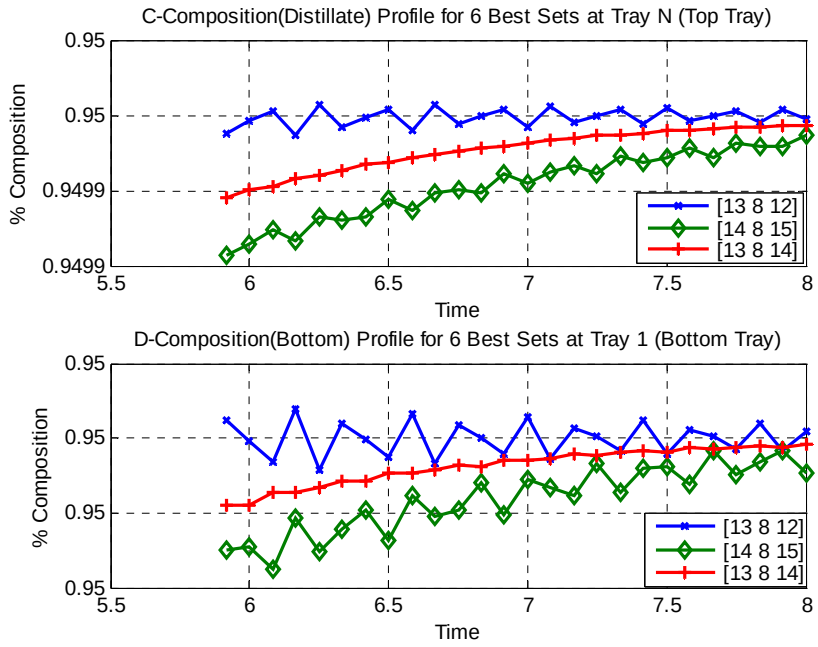

Figure 5.1 Selected Composition Profile for 6th -8th Hr of RDC operation

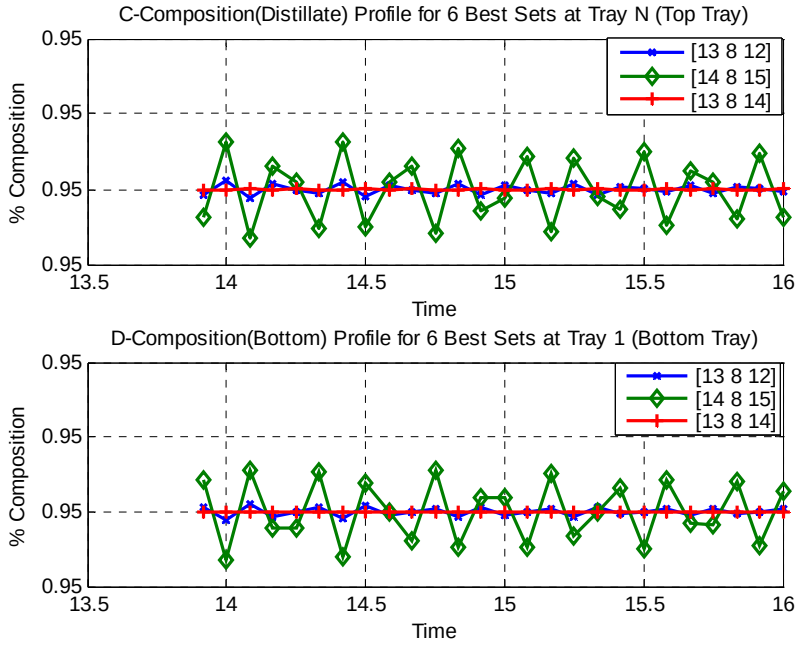

Figure 5.2 Selected Composition Profile for 14th to 16th Hr of RDC Operation

32 Hours: A third option was chosen to quadruple the initial time, i.e. to make the Process Run Time to 32 hours, it was seen that without doubt all the combinations were achieving steady state to an accuracy of 10^{-12}.

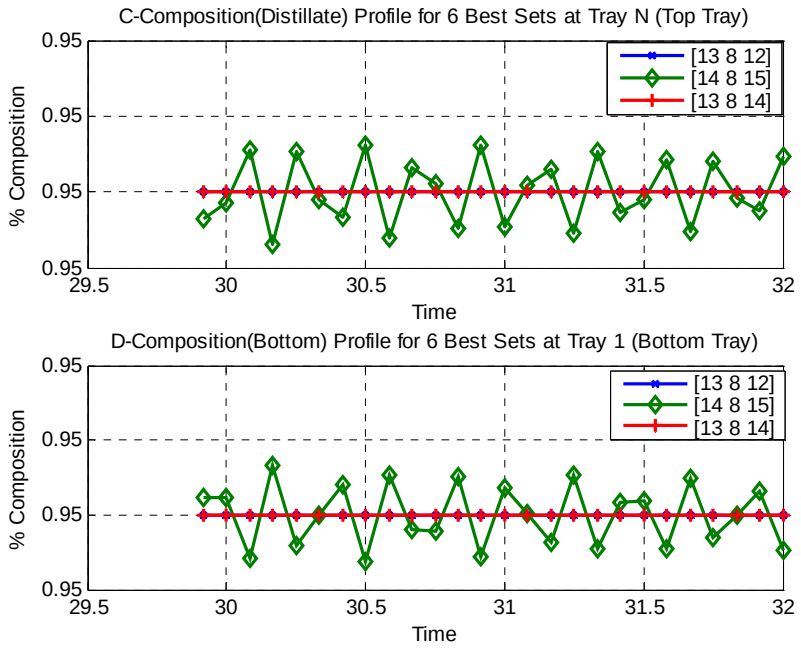

Figure 5.3 Selected Composition Profile for 30th -32nd Hr of RDC operation

Simulation time for each combination with process run time of 32 hours was found to be 105 seconds.

A minute and half per combination would mean 30 minutes per iteration based on a Swarm of size 20, and 100 iterations would mean a time of 50 hours. That is very huge time to dedicate. Although this computational time depends upon the type of CPU and

the RAM. In the best interest it was decided to use a Process Run Time of 16 hours without compromising the accuracy.

5.3 Objective Function

The cost function as explained in chapter 4.2 can be summarized as follows:

Objective 1: $V_S = f(N_S, N_{RX}, N_R) + k_1(95 - C_B) + k_1(95 - C_D)$

With Pressure = 9 bar, $N_{F1} = N_S + 2$ and $N_{F2} = N_S + N_{RX} + 1$

Objective 2: $V_S = f(N_S, N_{RX}, N_R, P) + k_1(95 - C_B) + k_1(95 - C_D)$

with Pressure = 9 bar,

Objective 3: $V_S = f(N_S, N_{RX}, N_R, P, N_{F1}, N_{F2}) + k_1(95 - C_B) + k_1(95 - C_D)$

Where k_1 is penalty factor with a very high value and C_B and C_D are concentrations of Bottom and Distillate products whose desired value is 95%. These objective functions are referred to as equality constraints wherein if a condition is not satisfied a high penalty is applied.

5.4 Algorithm

The algorithm, generalized for third objective, is started by initializing a large swarm consisting of the Stripping, Reactive, Rectifying Trays and Pressure. From the randomly generated swarm the location of feeds are generated using the stripping and reactive Trays the two swarms are merged to form one single swarm the rounding operation is performed on all but the pressure variable, initial values are determined for the RDC based on the swarm and the process is operated for 16 hours using each combination, the best among the swarm are selected after the cost function is evaluated to resize the swarm back to its original size. The velocity and positions of particles are updated using the HI-PSO and the algorithm is re-iterated until; the termination criterion is met.

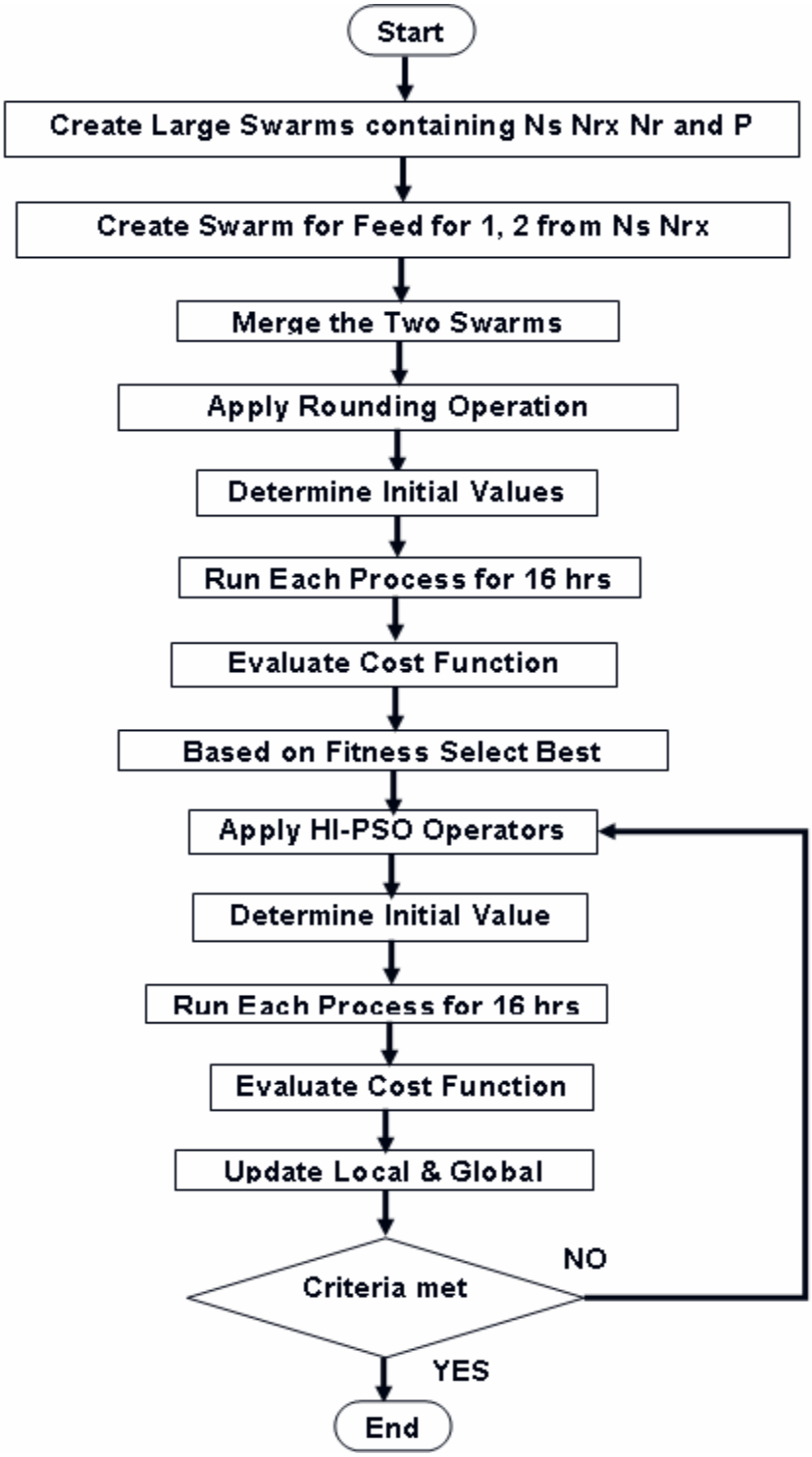

Figure 5.4: Hybrid Integer - PSO Algorithm

5.5 Results and Discussions

5.5.1 Optimization of Trays

The first objective of optimizing the trays in the Stripping, Reactive and Rectifying section by the use of HI-PSO has been realized. Shown in the figure 5.5 is the composition profile for the base case of [7 6 7] (N_S, N_{RX}, N_R) . The operating pressure is 9 bar and the feeds are located at locations 9 and 14 from the bottom, this is evident from the figure where sharp change in the profile can be seen. The base case has purity of both the products at 95%. This forms the basis of comparison to our achieved results.

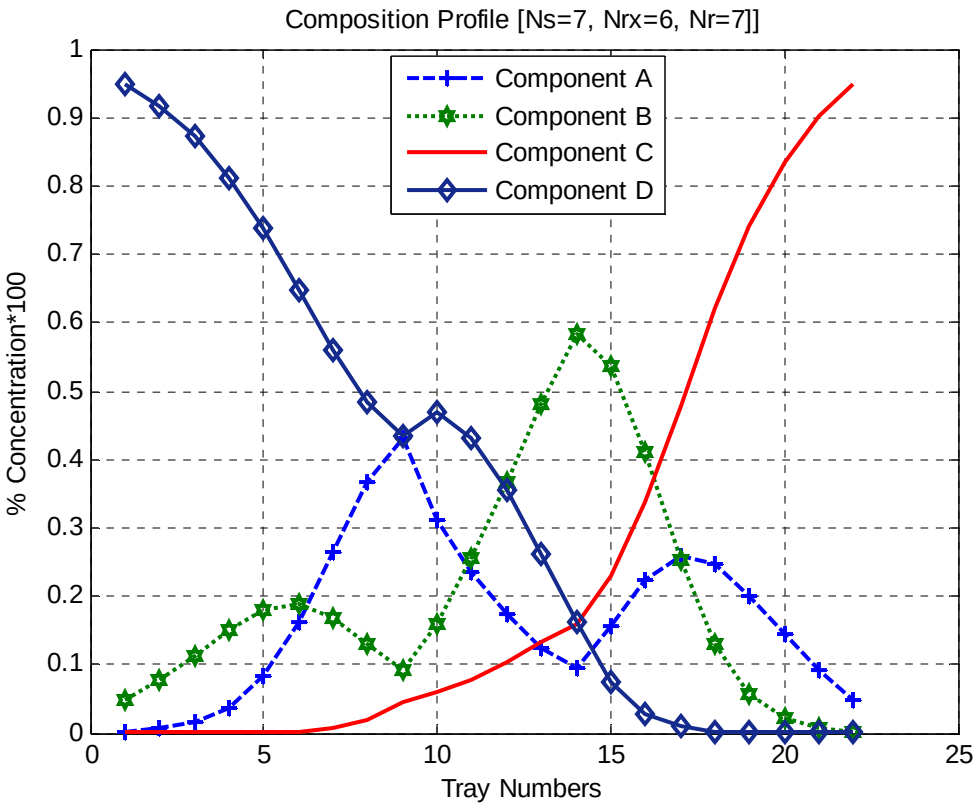

Figure 5.5 Composition Profile [7 6 7] Base Case

Using the Hybrid-Integer PSO we have optimized the vapor boilup and have found it to be 0.02683 compared to the base case with 0.0282. This gives a gain of 4.85% i.e. reduction cost of fuel.

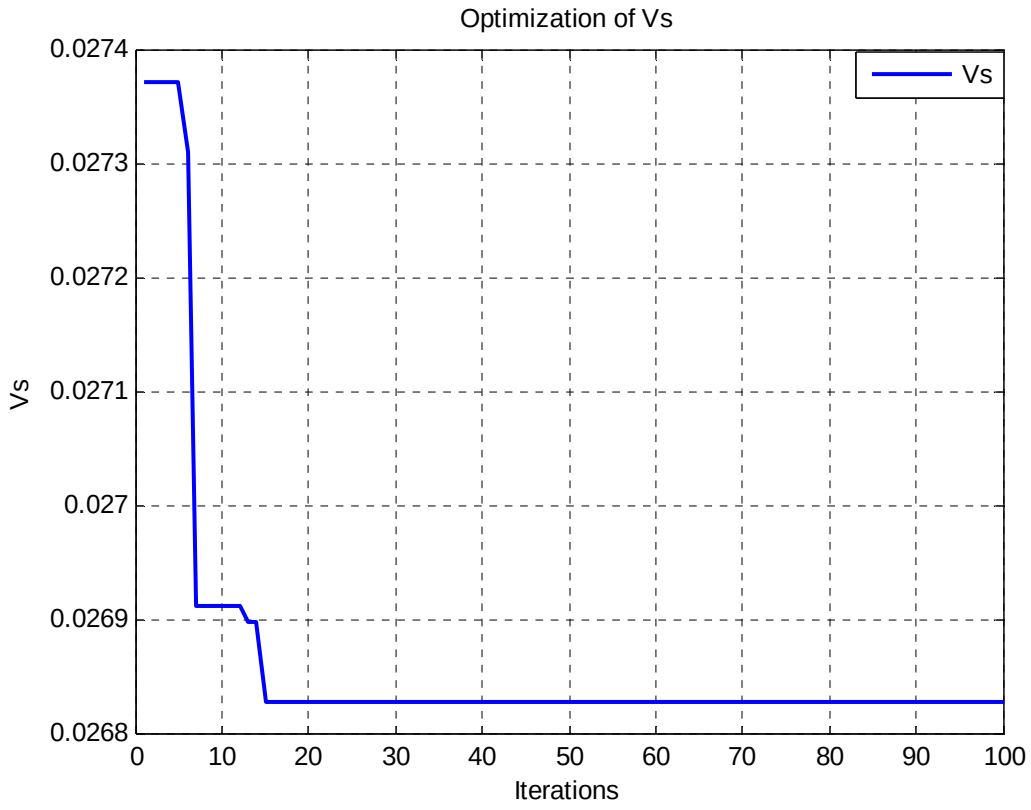

Figure 5.6 Optimization of Vs, Objective 1-Trays

The variation of V_s (Vapor boilup) under the optimization can be seen from the Figure 5.6. As can be seen the function minimizes around iteration 15 and remains steady thereof till the end of iterations. It is evident that the usage of Hybrid-Integer PSO has paid off well.

The optimal Tray configuration was found to be [13 8 12](N_S, N_{RX}, N_R). An increase in 7 trays in Stripping section, 2 trays in reactive section and 5 trays in rectifying section, totally an increase of height by 13 trays. It may strike as a surprise that instead of actually decreasing the cost of installation we have but increased the height which means an increase in initial investment and an increase in maintenance so on so forth. But this is not far sight. Although this fact cannot be ignored but more dominant among the fuel cost and the initial installation cost is fuel cost, fuel expended daily where as the installation is done one time. To conclude that we have traded off better profit would not be a biased conclusion.

Given in the figure 5.7 is the % concentration for both the Distillate and the Bottom products. It can be seen that the system has achieved steady state at around 4 hours and the concentration has been maintained as desired.

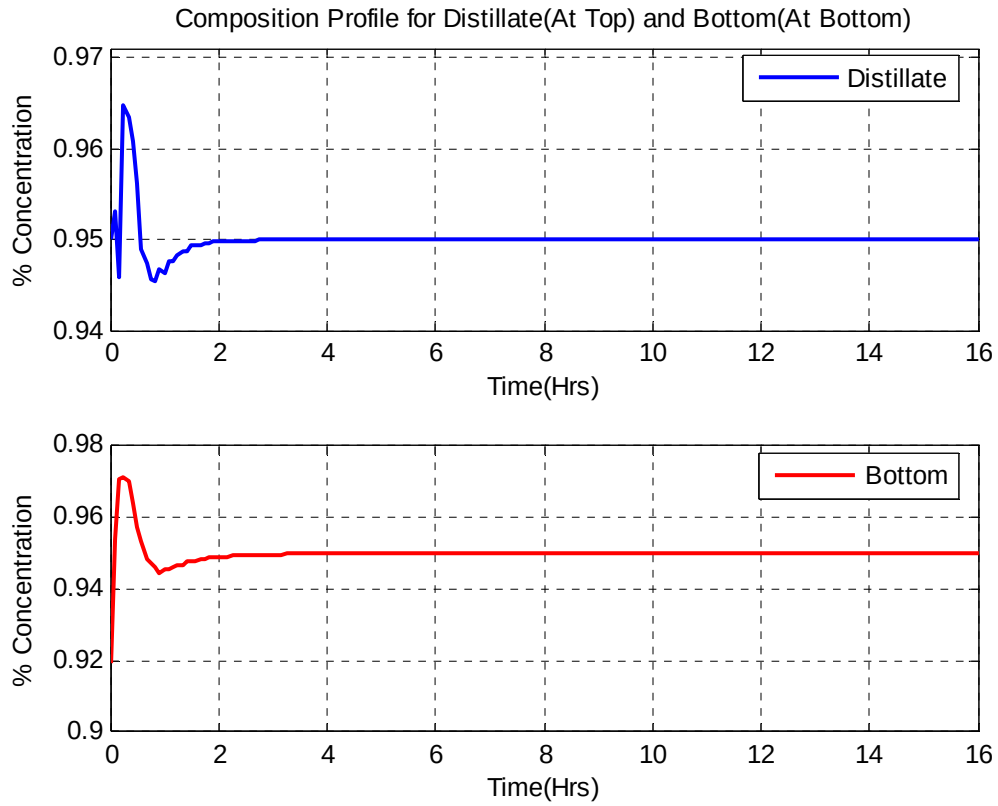

Figure 5.7 Composition Bottom and Distillate, Objective 1-Trays

Given in the Figure 5.8 is the composition profile of the optimal system as can be seen the system behaves similar to the base case with almost quick changes at the feed locations at 15 (N_S+2) and 22 (N_S+N_{RX}+1).

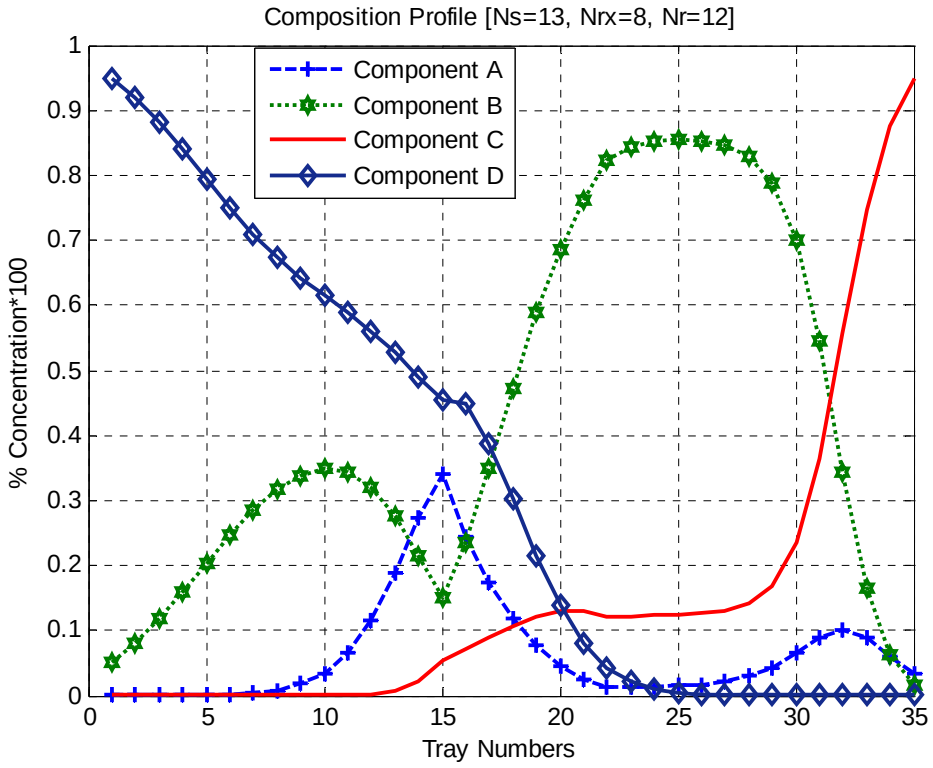

Figure 5.8 Composition Profile [13 8 12] Objective 1-Trays Optimization

The results of Hybrid-Integer PSO were compared with Simulations done by brute force (Exhaustive search) method i.e. evaluating every possible combination. 9261 combinations were tested and the minimum was found to be exactly the same as what was achieved by using HI-PSO. In the Table 5.1 Vs is the vapor Boilup R is the Reflux.

Table 5.1 Brute force Tray Optimization

Vs	R	Ns	Nrx	Nr
0.026829	0.03139	13	8	12
0.026913	0.031503	14	8	15
0.0269	0.031485	13	8	14
0.026855	0.031415	12	8	11
0.026887	0.031444	11	8	10
0.026925	0.03148	10	8	9

The important realization is expense of time. If we neglect the "No-Change" zone we find that HI-PSO for sure converges in around 20 iterations which would be 400 evaluations of the function. Where as the Brute Force method uses 9261 combination which is 23 times higher than that of HI-PSO. This shows the importance of optimization in real world problems and significance in large search space convergence.

5.5.2 Optimization of Trays and Column Pressure

The second objective of optimizing the trays in the Stripping reactive and rectifying section and the operating Pressure inside the Column has been realized by the use of HI-PSO. Shown in the figure 5.5 is the composition profile for the base case of [7 6 7] (N_S, N_{RX}, N_R) . The operating pressure is 9 atm and the feeds are located at locations 9 and 14, this is evident from the figure where sharp change in the profile can be seen. The base case has Concetration of both the products to be 95%. Again this forms the basis of comparison to our achived results.

In the second objective we relax the pressure as well using the settings described in the section 5.2.2.2 and retain the relaxation to the tray settings. The HI-PSO has achived more optimal solutions when extra variable was relaxed. Compared to the base case we have obtained a gain of 5.25 % and a gain of 0.4% over the first objective. The optimal V_S achived using HI-PSO for this objective was 0.02672

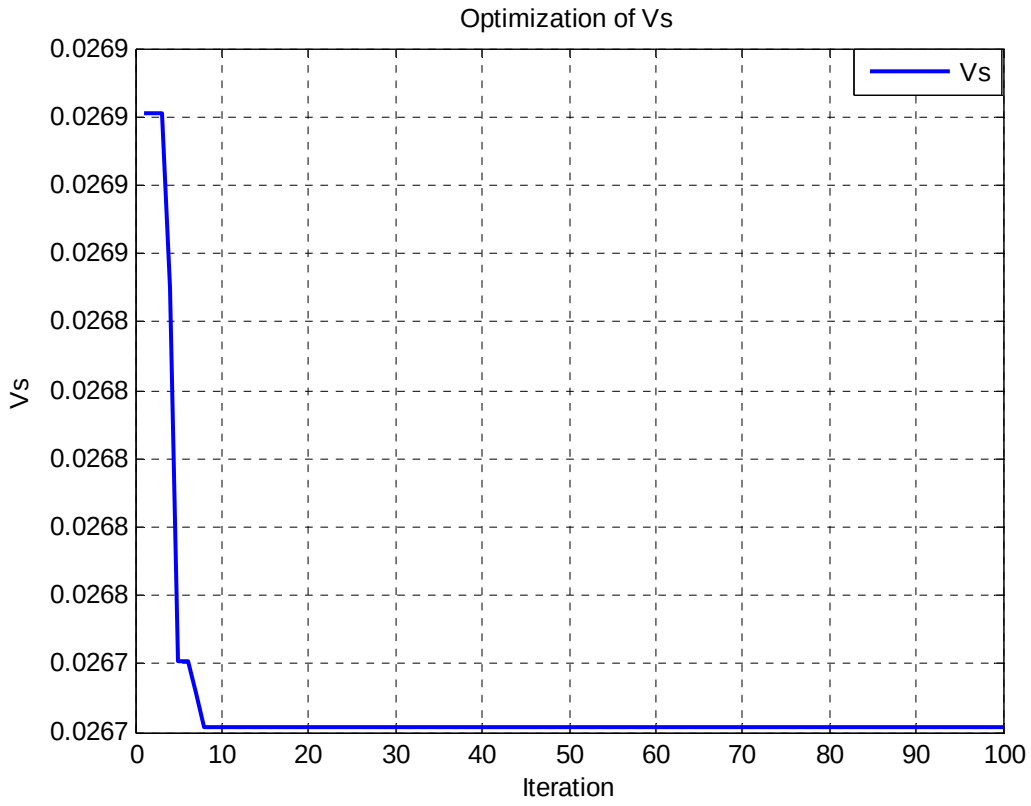

Figure 5.9 Optimization of Vs Objective 2–Trays and Column Pressure

The Figure 5.9 depicts the optimization of Vs when both the pressure and the trays are optimized. It was found that an optimal Pressure of 8.4 bar is sufficient in order to achieve the required concentration or purity of products. The optimal design by decrease in pressure can be attributed to increase in the number of trays in almost all sections by the same number as in the previous case. The HI-PSO converged to optimal solution in less than 10 iterations and remained steady thereof. The termination criterion for the optimization algorithm is the completion of all iterations.

Another to make sure that the optimal solution is not a local solution but a very accurate global solution we use the brute force method. Since the pressure was relaxed there was no choice but to make a partial brute force method based on analysis. Since a complete bruteforce would have no end to combinations, the pressure being a real variable and may assume any value between the two ends.

Hence a bruteforce method based on the analysis was taken into consideration. In this method we varied the combination in the following way.

- N_S – Stripping Section [10 to 14]
- N_{RX} – Reactive Section [6 to 10]
- N_R – Rectifying Section [11 to 15]
- P – Column Pressure [7.6, 8, 8.4, 8.8, 9.2]

In this way a total combination of 625 was obtained and we charted out the best 6 of those in the chart given below.

Table 5.2 Bruteforce - Tray and Pressure Optimization

Vs	R	Ns	Nrx	Nr	P
0.026722	0.031279	13	8	12	8.4
0.026745	0.031302	12	8	11	8.4
0.026772	0.031331	13	8	12	8.8
0.026778	0.031339	13	8	12	8.0
0.026797	0.031355	12	8	11	8.8
0.026798	0.031357	12	8	11	8.0

The next page shows the Concentration profiles for both the Bottom and the Distillate Product. As can be seen the HI-PSO provided with an accurate result even when a mixed integer was used among the variables. The trays are full integers and the Variable pressure is a real valued problem.

The figure 5.10 shows the concentration of the Distillate and Botton products and it can be easily seen that the system reaches the steady state with good accuracy around 95%.The simulation time for the system being 16 hours. The time is large enough for the system to settle to any slow settlings if present. A lesser time may seem viable, but in order to avoid an accidental miss of combinations which may gain steady state after a defined period and produce a better vapor boilup this time was used.

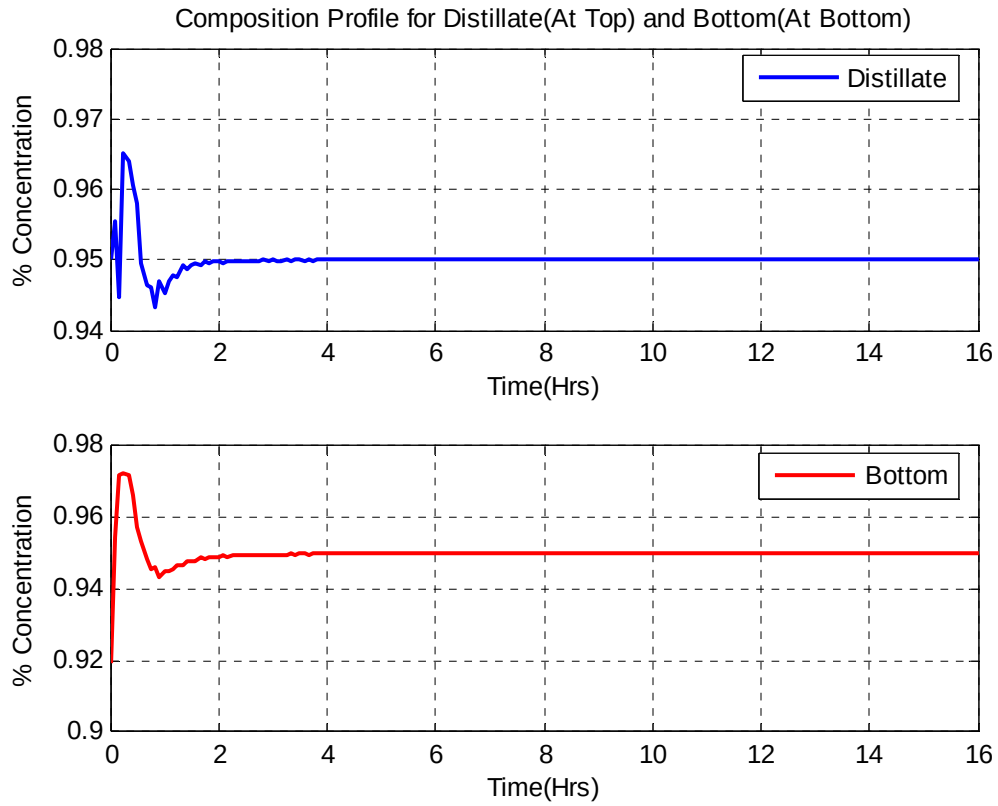

Figure 5.10 Products Purity Percentage Composition Objective 2–Trays and Column Pressure Optimization

In the figure 5.11 it can be seen that with decrease in pressure the reactions are actually taking place more accurately than in the previous case where the trays were optimized but the pressure was left un-optimized. To conclude we can say A drop in pressure can be compromised by increasing the number of trays in the rectifying and stripping section so as to attain optimal Vs.

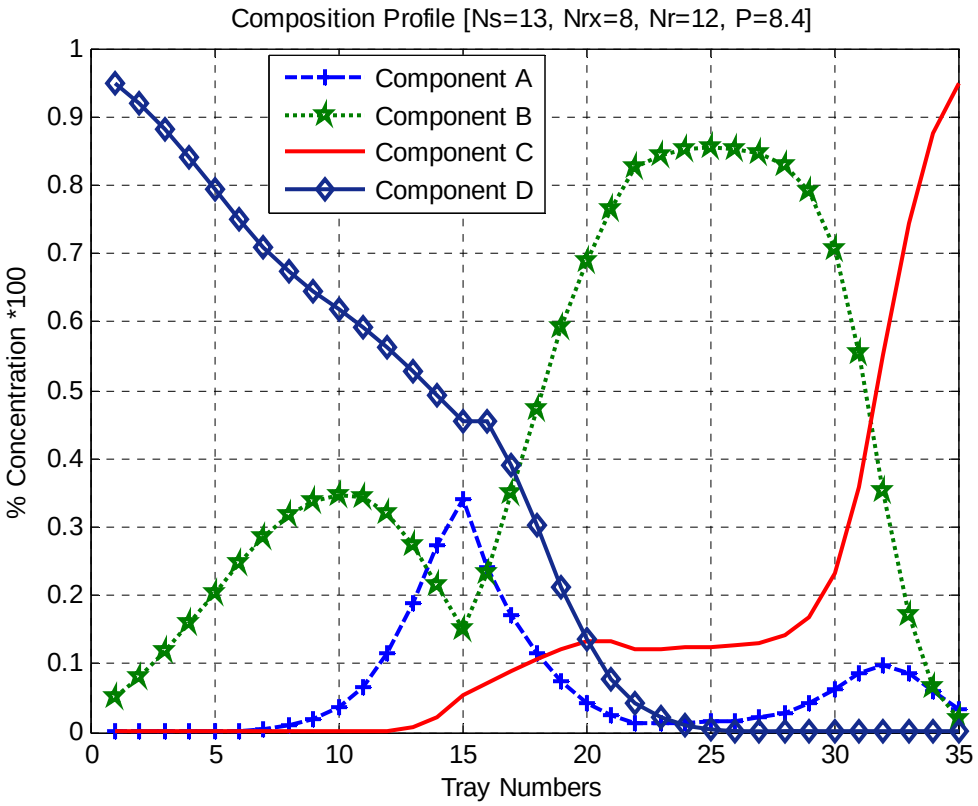

Figure 5.11 Composition Profile Objective 2 - Trays and Column Pressure Optimization

5.5.3 Optimization of Trays, Column Pressure and Feed Locations

The Third objective of optimizing the trays in the stripping reactive and rectifying section, the operating Pressure inside the Column and the feed locations of input feeds A and B has been achieved by the use of HI-PSO. For the base case of [7 6 7] (N_S, N_{RX}, N_R) . the operating pressure is 9 atm and the feeds are located at locations 9

and 14, the trays the pressure and the feeds are relaxed and optimized The base case has Concetration of both the products to be 95%.

The optimal design for the reactive distillation column was found to be

Trays: Ns = 11; Nrx =8; Nr=10

Pressure: 8.38 bar

Feeds: A = Ns+2; B = Ns+Nrx+3

Vs = 0.02661887748618

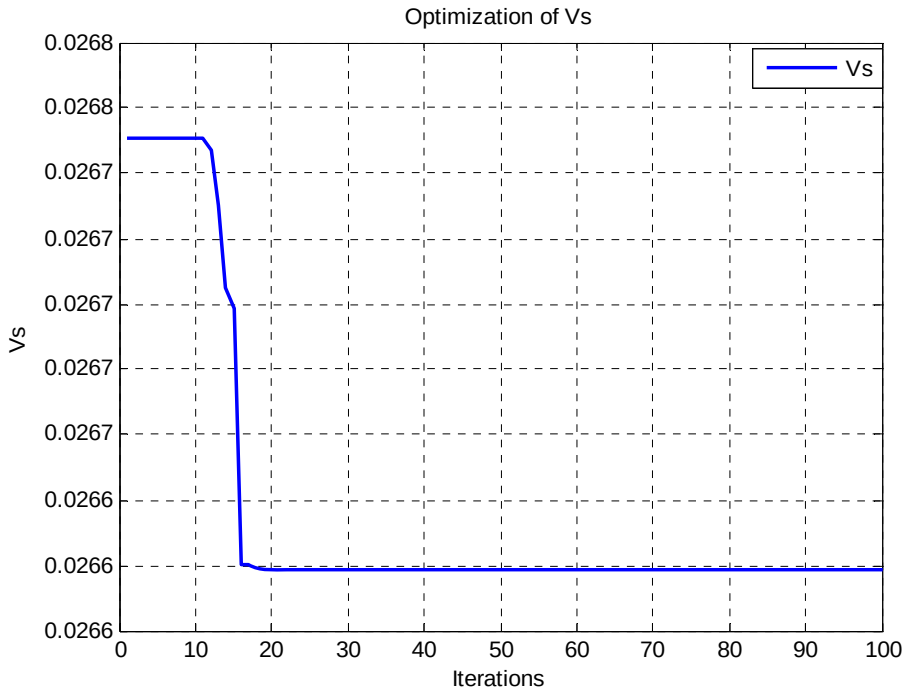

Figure 5.12 Optimization of Vs Objective 3 – Trays, Column Pressure and Feed Locations

HI-PSO has optimized the design of reactive distillation column and has saved 6% of cost in fuel consumption for the process compared to the base case of ideal RDC. Although the algorithm has taken a little longer than the previous two cases but still it is a best bet. This objective had 5 integer variables and one real variable. The algorithm converged to optimal solution in less than 20 iterations as can be seen in figure 5.12. This is a fast solution taking into account the time it would have taken if bruteforce method is applied to this problem. Based on the previous results we conclude that the HI-PSO converged onto better solution with faster pace.

Over all the optimal system height was increased by 9 trays, the pressure was dropped by around 0.62 bar and the feed location for feed 2 needed to be shifted up towards the 'TOP' in order to achieve the optimal Vs.

The composition profiles show that the system achieved steady state before 3 hours and remained steady there after. The concentration profiles show the characteristics of the system under the new design.

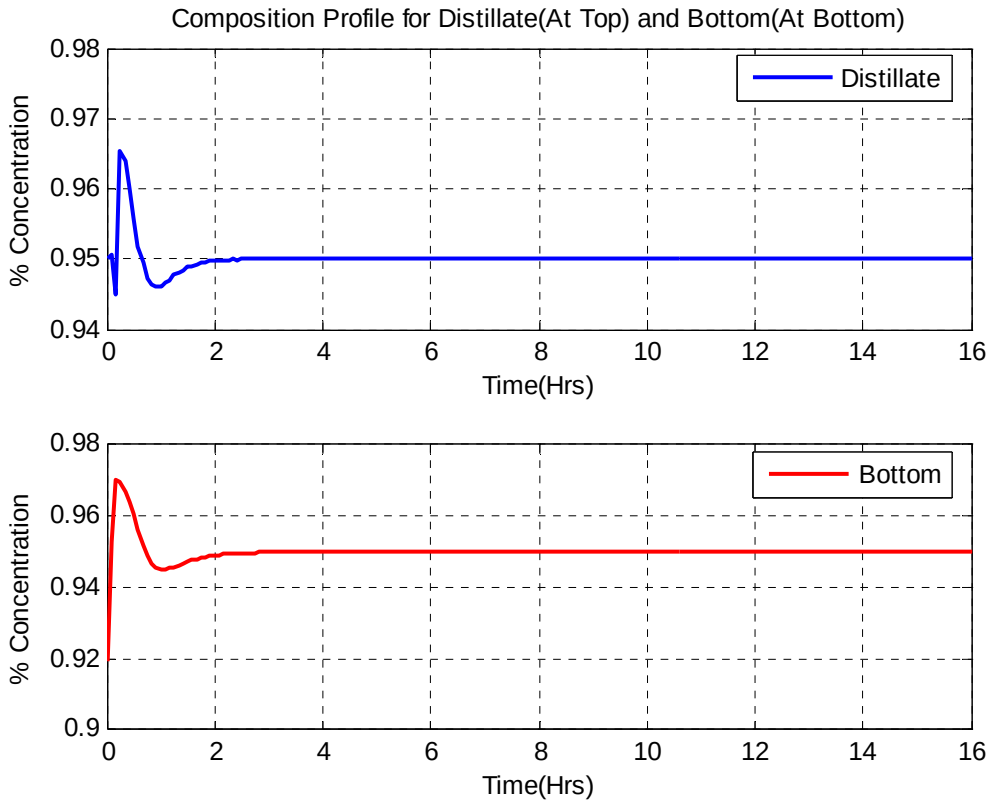

Figure 5.13 Products Purity Percentage Composition Objective 3 - Trays, Column Pressure and Feed Locations

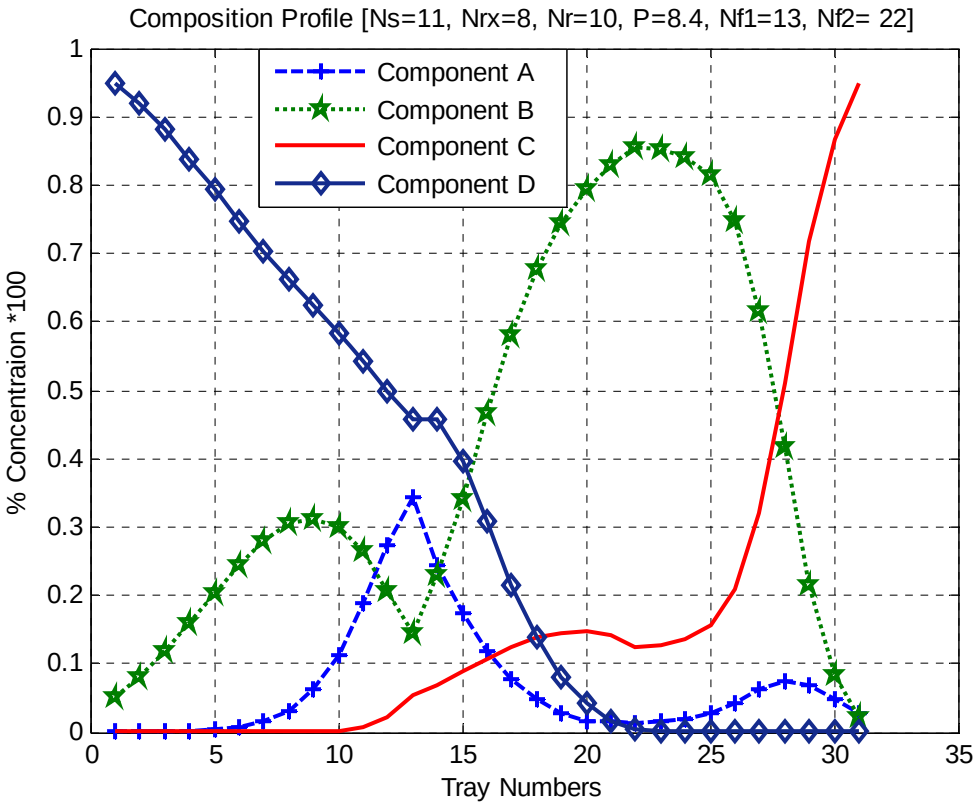

Figure 5.14 Composition Profile Objective 3 - Trays, Column Pressure and Feed Locations

The figure 5.14 shows the composition profile for the optimal design as calculated by the HI-PSO. Thus Proper choice of feed locations if optimal can really reduce the size of the column and decrease Vs too.The tables 5.3 and 5.4 show the comparison of the 20 Best and worst cases encountered by HI-PSO in the first 20 iterations.

Comparison of Worst 20 cases evaluated By HI-PSO

Table 5.3 Tray + Pressure + Feed Worst Cases

Vs	R	Ns	Nrx	Nr	P	Feed-1	Feed2
91.057	0.065935	20	5	25	14.863	23	28
81.057	0.065935	25	5	25	15	28	29
108.06	0.065935	25	23	25	15	32	51
91.057	0.065935	24	19	18	12.592	30	35
88.057	0.065935	19	5	25	12.422	21	27
64.057	0.065935	16	20	12	11.056	26	39
56.057	0.065935	20	6	25	5.8231	18	24
125.06	0.065935	24	6	25	15	21	32
118	0.065935	25	5	25	15	22	31
129	0.065935	22	5	25	12.326	23	30
56.057	0.065935	21	6	25	8.4408	23	25
69.057	0.065935	25	7	13	14.638	28	29
95.057	0.065935	19	5	24	15	21	27
125.06	0.065935	25	5	25	14.867	28	33
111.01	0.004587	25	12	25	5	22	40
69.057	0.065935	25	7	25	13.113	28	35
116	0.065935	18	7	25	15	22	28
163.06	0.065935	24	25	25	11.168	27	52
122.06	0.065935	20	5	25	11.461	21	25
40	0.065935	25	5	25	12.667	22	29

Comparison of Best 20 cases evaluated By HI-PSO

Table 5.4 Tray + Pressure + Feed Best Cases

Vs	R	Ns	Nrx	Nr	P	Feed-1	Feed-2
0.026751	0.031308	12	8	11	8.5235	14	21
0.027373	0.031971	12	7	11	8.3137	14	22
0.027397	0.032023	11	7	12	8.8549	13	21
0.027428	0.032023	10	7	9	8.2801	12	19
0.032455	0.037088	11	6	11	8.2002	12	20
0.028117	0.032713	13	7	11	8.3852	15	21
0.027541	0.032179	12	6	13	8.4877	14	20
0.027727	0.032362	10	6	11	8.5638	12	19
0.027179	0.031796	12	8	12	8.6688	14	21
0.027347	0.031983	12	7	12	8.7035	14	20
0.026752	0.031309	12	8	11	8.5335	14	21
0.026747	0.031304	12	8	11	8.3035	14	21
0.02673	0.031276	11	8	10	8.0739	13	21
0.026705	0.03125	11	8	10	8.233	13	21
0.026698	0.031243	11	8	10	8.4496	13	21
0.02662	0.031155	11	8	10	8.4612	13	22
0.026621	0.031156	11	8	10	8.4811	13	22
0.026619	0.031154	11	8	10	8.4273	13	22
0.026619	0.031153	11	8	10	8.4007	13	22
0.026619	0.031153	11	8	10	8.3895	13	22
0.026619	0.031153	11	8	10	8.39	13	22

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

Integer PSO can handle integer problems very well, but the main problem encountered in Integer PSO is that the convergence rate is not very fast although it's good. In systems like RDC which is highly non-linear, takes a lot of time to simulate. Large number of iterations would mean large amount of time. Hybrid PSO with breeding and sub-population has the ability to enhance the convergence rate very fast and obtain the global minima without getting stuck in local minima. A hybrid-Integer PSO resulting from Large Initial Swarm, Integer PSO and Hybrid PSO thus provides faster convergence in small number of iterations to problems like that of RDC. HI-PSO can be applied to a wide variety of problems ranging from simple mathematical to complex and hard industrial problems which are integer based. The development of Hybrid Integer PSO has added to the results and has proved to be more powerful than the standard integer PSO. An ideal Reactive distillation column can be optimized by finding the optimal number of trays in the stripping, reactive and rectifying section along with the pressure in the column and proper feed locations.

An increase in trays for an optimal design provided the fuel factor Vs is low is not a bad compromise to make. The height of the tower or the installation of trays being a one time installation may look like a gross input. But the savings made on fuel of about 6% per hour does really prove to be significant.

A drop in pressure can be compromised by increasing the number of trays in the rectifying and stripping section so as to attain optimal Vs.

Proper choice of feed locations if optimal can really reduce the size of the column and decrease Vs too.

6.2 Recommendations for Future Work

- HI-PSO can be used in parallel environment where every particle can be evaluated separately.
- HI-PSO may be coded for distributed environment where there exists a Master-Slave network.
- With in the search space the distributed PSO can greatly effect the time of simulation, hence a cluster of PC's can be used to serve the purpose of distributed PSO.

- A generic algorithm can be designed based on the principle of chemical engineering to find the initial values with using the method of extrapolation.
- Effect of optimizing the Tray + Feed and Feed + Pressure can also be investigated although may sound simple, but this may give some more insight on the behavior of RDC.
- Other variables like the molar holdup, height of the tower can be introduced as optimization parameters.

REFERENCES

1. Edgar, Himmelblau and Lasdon “Optimization of Chemical Processes” McGraw-Hill 2001
2. K. Sundmacher and A. Kienle, "Reactive distillation: status and future directions." Weinheim: WILEY-VCH, 2003
3. Al-Arfaj, M.A & Luyben, W. L. “Effect of Number of Fractionating Trays on Reactive Distillation Performance,” AIChE Journal; v46; 2417-2425; 2000
4. Ciric, A. R., & Gu, D. (1994). Synthesis of nonequilibrium reactive distillation processes by MINLP optimization. *AIChE Journal*, *40*(9), 1479.
5. Z. Michalewicz, and D. B. Fogel, “How to Solve It: Modern Heuristics”, Springer-Verlag Berlin Heidelberg New York (2002)
6. J. H. Holland, “Adaptation in Natural and Arti_cial Systems”, University of Michigan Press, Ann Arbor, 1929.
7. T. Back, D. B. Fogel, and Z. Michalewicz, “Handbook of Evolutionary Computation”, Oxford University Press, (1997).
8. S. Hackwood, and G. Beni, “Self-Organization of Sensors for Swarm Intelligence”, In Proceedings IEEE 1992 International Conference on Robotics and Automation, Los Alamitos, CA: IEEE Computer Society Press, (1992):819-829.
9. E. Bonabeau, M. Dorigo and G. Theraulaz, “Swarm Intelligence: “From Natural to Arti_cial Systems”, Oxford University Press Inc., ISBN 0-19-513158-4 (cloth); ISBN 0-19-513159-2 (pbk.), (1999): 1-22.

10. M. F. Cardoso, R. L. Salcedo, S. Feyo de Azevedo, D. Barbosa “Optimization of reactive distillation processes with simulated annealing” Chemical Engineering Science 55 (2000) 5059-5078
11. J. Kennedy, and R. C. Eberhart, “Swarm Intelligence”, ISBN 1-55860-595-9, Academic Press (2001).
12. J. Kennedy, and R. C. Eberhart, “Particle swarm optimization”,Proceedings of the 1995 IEEE International Conference on Neural Networks,vol. 4, IEEE Press, (1995): 1942-1948.
13. P. J. Angeline, “Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences”, Evolutionary Programming VII, Lecture Notes in Computer Science 1447, Springer, (1998):601-610.
14. J. Kennedy, “Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, IEEE Press, (1999): 1931-1938.
15. M. Clerc, and J. Kennedy, “The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space”, IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, February, (2002): 58-73.
16. M. Løvbjerg, T. K. Rasmussen, and T. Krink, “Hybrid Particle Swarm Optimiser with Breeding and Subpopulations”, Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001).

17. A. Carlisle, and G. Dozier, “An off-the-shelf PSO”, Proceedings of the workshop on particle swarm optimization, Purdue school of engineering and technology, Indianapolis, IN, (2001).
18. Y. Shi and R. C. Eberhart, “Parameter Selection in Particle Swarm Optimization”, Evolutionary Programming VII, Springer, Lecture Notes in Computer Science 1447, (1998): 591-600.
19. Y. Shi, and R. C. Eberhart, “Empirical Study of Particle Swarm Optimization”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, IEEE Press, (1999): 1945-1950.
20. M. A. Al-Arfaj and W. L. Luyben, "Comparison of alternative control structures for an ideal two-product reactive distillation column," *Ind. Eng. Chem. Res*., vol. 39, pp. 3298-3307, 2000.
21. Parsopoulos, K.E., Vrahatis, M.N. “On the computation of all global minimizers through particle swarm optimization”. IEEE Trans. Evol. Comp. 8 (2004) 211–224
22. K. Sundmacher and A. Kienle, "Reactive distillation: status and future directions." Weinheim: WILLEY-VCH, 2003.
23. R. Taylor and R. Krishna, "Modeling reactive distillation," *Chem. Eng. Sci*., vol. 55, pp. 5183-5229, 2000.
24. M. F. Doherty and G. Buzad, "Reactive distillation by design.," *Transactions of the Institution of Chemical Engineers, Part* A, vol. 70, pp. 448-458, 1992.

25. Jose Leboreiro, Joaquin Acevedo “Processes synthesis and design of distillation sequences using modular simulators: a genetic algorithm framework” Computers and Chemical Engineering 28 (2004) 1223–1236

26. Kian Huat Low, Eva Sorensen “Simultaneous optimal design and operation of multipurpose batch distillation columns” Chemical Engineering and Processing 43 (2004) 273–289

27. G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd (Eds.), Handbooks in OR & MS, Vol. 1: Optimization, Elsevier, 1989.

28. V.P. Plagianakos and M.N. Vrahatis, “Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2000), Como, Italy, 2000.

29. S.S. Rao, Engineering Optimization:Theory and Practice, Wiley Eastern: New Delhi, 1996.

30. D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press: New York, 1995.

31. Y. Shi and R.C. Eberhart, “Parameter Selection in Particle Swarm Optimization", EvolutionaryProgramming VII, pp. 591-600, 1998.

32. H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, 1995.

33. G. Rudolph, “An Evolutionary Algorithm for Integer Programming",Y. Davidor, H. -P. Schwefel, R. Manner (Eds.), Parallel Problem Solving from Nature 3, pp. 139-148,Springer, 1994.

34. D.A. Gall, “A Practical Multifactor Optimization Criterion", A. Levi, T.P. Vogl (Eds.), Recent Advances in Optimization Techniques, pp. 369-386, 1966.
35. R.C. Kelahan and J.L. Gaddy, “Application of the Adaptive Random Search to Discrete and Mixed Integer Optimization", International Journal for Numerical Methods in Enginnering, Vol. 12, pp. 289-298, 1978.
36. E C Laskari, K E Parsopoulos and M N Vrahatis “PSO for integer programming” IEEE transactions on power system February 2002.
37. W. M. Spears, ”Simple subpopulation schemes”, Proceedings of the Evolutionary Programming Conference 1994, pp. 296-307.
38. Sanne Melles, Johan Grievink, Stany M. Schrans “Optimisation of the conceptual design of reactive distillation columns” Chemical Engineering Science 55 (2000) 2089}2097
39. A. Glankwahmdee, J.S. Liebman and G.L. Hogg, “Unconstrained Discrete Nonlinear Programming", Engineering Optimization, Vol. 4, pp. 95-107, 1979.
40. G. Rudolph, “An Evolutionary Algorithm for Integer Programming", Y. Davidor, H. P. Schwefel, R. Manner, Parallel Problem Solving from Nature 3, pp. 139-148, Springer, 1994.
41. Olanrewaju Moshood Jide “Development and Application of Linear State Estimators in Control of Reactive Distillation”, King Fahd University of Petroleum & Minerals, May 2005

VITAE

Name:	Mohammed Mirza Amer Baig
Education:	*Master of Science (M.S)* Electrical Engineering King Fahd University of Petroelum and Minerals
	Bachelor of Engineering (B.E) Electr. & Comm. Engg. Gulbarga University, India
Date of Birth:	1st August 1978
Contact:	amerbaig1978@yahoo.com