METAHEURISTIC APPROACHES TO THE POOLING PROBLEM

by

Gökalp Erbeyoğlu

B.S., Industrial Engineering, Boğaziçi University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University
2013

METAHEURISTIC APPROACHES TO THE POOLING PROBLEM

APPROVED BY:

Prof. Ümit Bilge (Thesis Supervisor)	....
Prof. Tülin Aktin	....
Assoc. Prof. Ali Tamer Ünal	....

DATE OF APPROVAL: . .2013

ACKNOWLEDGEMENTS

First of all, I would like to thank Prof. Ümit Bilge for her endless support, patience and guidance throughout my entire graduate study. This work would not be completed without her wise advices and suggestions. Secondly, I would like to express my gratitudes to Prof. Tülin Aktin, Prof. Necati Aras and Assoc. Prof. Ali Tamer Ünal for taking part in my thesis jury and providing valuable comments.

I would like to thank Erinç, Umut and Mehmet for their friendship and invaluable support during this study and for the great times we spent together in BUFAIM Laboratory. I also want to thank my colleagues Ezgi, Turgut, Burak, Merve and Mert for their friendship.

I would like to thank my cousins Burak and Mehmet, and my friends from the first days of university, Burak and Mustafa, for the times we spent together.

I will never be able to find correct words to express my gratitudes to my family. I would like to thank my brother for trusting and encouraging me at times when I feel doubtful about my decisions. I thank my mother for her unlimited encouragement and my father for giving me perspective whenever I needed. I am sure that things would be much harder if I did not have their support and love.

I also thank TUBITAK for their financial support during my graduate study.

ABSTRACT

METAHEURISTIC APPROACHES TO THE POOLING PROBLEM

The pooling problem, which has several application areas in chemical industry, is an extension of the blending problem and aims to find the optimal composition of materials in a two-stage network while obeying quality limitations for the end products. The pooling problem has a bilinear structure and it is NP-hard. The exact methods to solve the pooling problem are inefficient for large instances and a few heuristic methods exist. In this thesis, our aim is to propose two metaheuristic methods that are based on particle swarm optimization (PSO) and simulated annealing (SA). Both of the proposed approaches take advantage of the bilinear structure of the problem. For PSO-based method, a search variable is selected among the variable sets causing bilinearity and subjected to particle swarm optimization. For SA-based procedure, a variable neighboring scheme that is similar to a previously used one for the pooling problem is employed. Extensive experiments are conducted to evaluate the performances of these methods and they indicate the success of the proposed solution methods.

ÖZET

HAVUZLU MALZEME KARIŞIMI PROBLEMİNE SEZGİSEL YAKLAŞIMLAR

Havuzlu malzeme karışımı problemi kimya endüstrisinde çeşitli uygulama alanlarına sahiptir ve harmanlama probleminin bir uzantısıdır. Hammaddelerin iki aşamalı bir ağda, kalite kısıtlamalarına uyarak en iyi karıştırılma oranlarını bulmayı hedefler. Havuzlu malzeme karışımı problemi ikili-doğrusal bir yapıdadır ve NP-zor olarak sınıflandırılır. Eniyileyen sonucu garanti eden çözüm yöntemleri büyük boyutlu problemler için yetersiz kalmaktadır ve problemin çözümü için az sayıda sezgisel yöntem uygulanmıştır. Bu çalışmada amacımız, parçacık sürü eniyilemesi ve benzetimli tavlama tabanlı iki sezgisel yöntem önermektir ve bu iki yöntem de problemin ikili-doğrusal yapısından faydalanmaktadır. Parçacık sürü eniyilemesi tabanlı yöntemde ikili-doğrusallığa sebep olan değişken kümelerinden bir tanesi seçilmiş ve üzerinde parçacık sürü eniyilemesi yöntemi uygulanmıştır. Benzetimli tavlama tabanlı yöntemde ise, literatürde uygulanmış bir yöntem esas alınarak bir değişken komşuluk tanımı uygulanmıştır. Önerilen yöntemlerin başarısını değerlendirmek için uygulanan kapsamlı testler, uygulanan yöntemlerin başarılı olduğuna işaret etmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_i^L	Minimum required usage of raw material i
A_i^U	Maximum availability of raw material i
c_{ij}	Unit cost of raw material i sent to output j
C_{ik}	Level of quality attribute k in raw material i
c_{il}	Unit cost of raw material i sent to pool l
$cLen$	Cycle length
D_j^L	Minimum required production of end product j
D_j^U	Demand upper limit of end product j
d_j	Unit revenue for product j
$F(S)$	Objective function value of solution S
FAR	Final acceptance ratio for simulated annealing
FCL	Final cycle length for simulated annealing
FNS	Final neighborhood size for simulated annealing
$globBest$	Best position vector of the population
I	Set of input streams (raw materials)
IAR	Initial acceptance ratio for simulated annealing
ICL	Initial cycle length for simulated annealing
INS	Initial neighborhood size for simulated annealing
J	Set of output streams (end products)
K	Set of quality attributes
k_{max}	Maximum neighborhood size
L	Set of pools
LB_{STP}	Lower bound found by solution of STP-formulation
$maxIter$	Maximum number of iterations
$nSize$	Current neighborhood size
P	Position vector of a particle
P_{jk}^L	Lower limit of quality attribute k in end product j
P_{jk}^U	Upper limit of quality attribute k in end product j

p_{lk}	Level of quality attribute k in pool l
$particleBest$	Best position vector of a particle
$popSize$	Population size
q_{il}	Proportion of flow from input stream i to pool l among all flows into pool l
S	Current solution
S'	Neighboring solution
S_l	Capacity of pool l
T	Temperature
t_{lj}	Proportion of flow from pool l to output j
T_0	Initial temperature
T_f	Final temperature
T_X	Set of (i, l) pairs for which input to pool connection exists
T_Y	Set of (l, j) pairs for which pool to output connection exists
T_Z	Set of (i, j) pairs for which input to output connection exists
V	Velocity vector of a particle
V_{max}	Maximum velocity for particle swarm optimization
x_{il}	Flow from input stream i to pool l
y_{lj}	Flow from pool l to output j
z_{ij}	Flow from input stream i to output j
α	Cooling coefficient
β	Initial neighborhood size coefficient
γ	Neighborhood update coefficient
δ	Subspace indicator
$\overline{\lvert\Delta f_0\rvert}$	Average of individual absolute differences in fitness values of the solution pairs at initial temperature
$\overline{\lvert\Delta f_f\rvert}$	Estimated average of individual absolute differences in fitness values of the solution pairs at final temperature
θ	Cycle length update coefficient
ϕ_1	Upper bound on random movement towards personal best of a particle

ϕ_2	Upper bound on random movement towards population best particle
$\phi_{1,initial}$	Initial value of upper bound on random movement towards personal best of a particle
χ	Constriction coefficient
ω	Inertia weight
$\omega curv$	Inertia weight update curvature value
ω_{final}	Final value of inertia weight
$\omega_{initial}$	Initial value of inertia weight

LIST OF ACRONYMS/ABBREVIATIONS

ALT	Alternate Heuristic
BLP	Bilinear Programming
GCH	Greedy Construction Heuristic
LP	Linear Programming
MALT	Multi-start Alternate Heuristic
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SLP	Successive Linear Programming
VNS	Variable Neighborhood Search

1. INTRODUCTION

The blending problem in linear programming (LP) is a classical problem concerning mixing a set of raw materials with given quality attributes into final products with predefined quality limitations while profit is maximized. The pooling problem, which is the topic of this thesis, is an extension to the blending problem. While the blending problem finds the optimal composition of materials by directly mixing the raw materials into final products, the pooling problem incorporates intermediate pools where intermediary mixes can be prepared out of raw materials and the output mixes are formed by blending the feeds from pools as well as the raw materials. The pooling problem has several application areas in chemical industry, such as petroleum refining, natural gas transportation and wastewater treatment (Misener and Floudas, 2009).

The formulation of the standard pooling problem has a feed-forward network topology and there are three layers in the network: firstly, raw materials or input streams that can feed into both intermediate pools and end product mixes, secondly intermediate pools which can have multiple incoming and outgoing arcs and lastly end product pools that produce final blends as the output of the system (Audet *et al.*, 2004; Greenberg, 1995; Misener and Floudas, 2009). Figure 1.1 depicts the general structure of the problem where triangles are used to represent input streams, circles for pools and squares for end products. While the blending problem can be formulated as an LP, the intermediate pools in the pooling problem disturb the linearity of the problem at different degrees.

The pooling problem can be categorized into five classes as the standard pooling problem, the generalized pooling problem, the extended pooling problem, the nonlinear blending problem and the crude oil operations. The standard pooling problem optimizes the flows between sources, intermediate pools and output pools subject to quality constraints with linear blending assumption. The generalized pooling problem allows between-pool flows. The extended pooling problem integrates relevant legislative

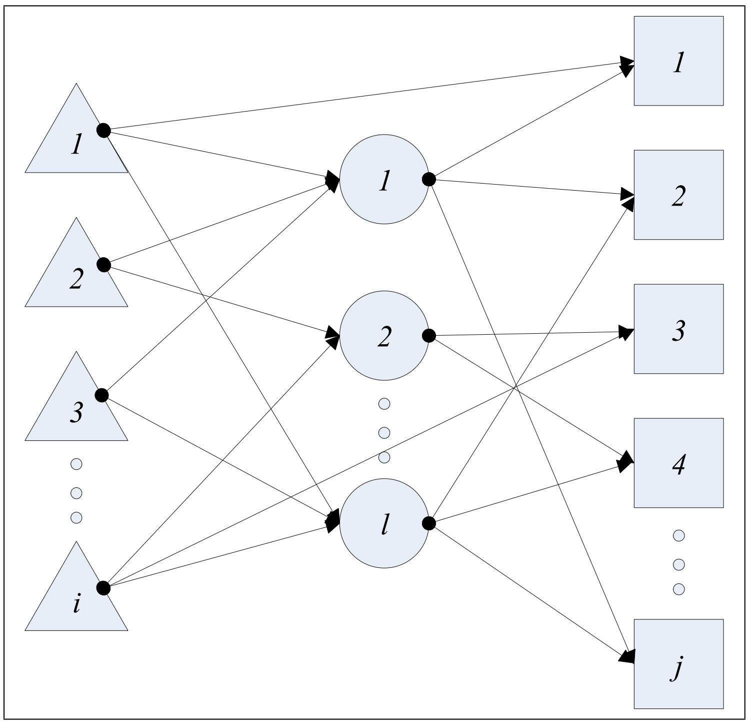

Figure 1.1. The General Structure of the pooling problem.

bounds into constraint sets. The nonlinear blending problem incorporates nonlinear blending rules instead of assuming linear blending. The crude oil operations deal with the front-end of a refinery plant and add a scheduling component to the problem. The standard pooling problem can be formulated as a bilinear program. On the other hand, the generalized pooling problem can be modeled as mixed-integer bilinear program, the nonlinear blending program can be modeled as a nonconvex nonlinear program, and the extended pooling problem and the crude oil operations are modeled as mixed-integer nonconvex nonlinear problems (Misener and Floudas, 2009). In this study, the standard pooling problem is considered and it is mentioned as the pooling problem throughout the study.

Pooling problem can be summarized as the following: There is a set of input streams and each raw material has a set of quality attributes. Also, there are upper and lower limits on these quality attributes that should be satisfied for the end products. When different products are mixed in intermediate pools and final product pools, it is assumed that the quality attributes of the output are calculated as the weighted average of the quality attributes of the incoming materials by their volume (Audet *et al.*, 2004; Greenberg, 1995; Gupte *et al.*, 2012; Misener and Floudas, 2009).

Pooling problem belongs to the class of bilinear programming problems (BLP), which is a subset of nonconvex quadratic program. In BLPs, nonlinearities occur such that the optimization problem reduces to an LP if one of the two variable sets that cause nonlinearities is fixed. The most general form of BLPs is given in Equations 1.1, 1.2 and 1.3 (Al-Khayyal, 1992). BLPs can be classified as strongly NP-Hard problems since it includes NP-hard linear maxmin problem (Audet *et al.*, 2004).

$$min \qquad c_0^T x + x^T A_0 y + d_0^T y \tag{1.1}$$

$$s.t. \qquad c_i^T x + x^T A_i y + d_i^T y \leq b_i \forall i \tag{1.2}$$

$$(x, y) \in S = \{(x, y) : Cx + Dy \leq b, x \geq 0, y \geq 0\} \tag{1.3}$$

In this thesis, pooling problem is examined and two metaheuristic approaches, namely particle swarm optimization and simulated annealing, are adapted for the problem. In order to evaluate the performances of these methods, a number of problem instances in the literature are experimented and some new instances are fabricated while general purpose solvers and heuristic procedures in the literature are used as benchmark methods.

The rest of this thesis is organized as follows: Chapter 2 presents a literature review on pooling problem, particle swarm optimization (PSO) and simulated annealing (SA) concepts. In Chapter 3, the two proposed solution methods are presented in detail. Chapter 4 covers the numerical experiments and comparisons of proposed methods. In the last chapter, Chapter 5, concluding remarks are provided.

2. LITERATURE REVIEW

This chapter provides a brief literature survey on the pooling problem in Section 2.1, its formulations and solution methods. Then, Section 2.2 and Section 2.3 outline particle swarm optimization and simulated annealing concepts that are used in this thesis.

2.1. Pooling Problem

2.1.1. Model Formulations

There are several different formulations for the pooling problem that are mathematically equivalent but varying in terms of relaxation tightness and problem size. Haverly (1978) formulated the original pooling problem, denoted as P-formulation. This formulation explicitly keeps track of flows between raw materials to pools and end products, and flows between pools and end products. Ben-Tal *et al.* (1994) proposed the Q-formulation, which replaces the variables that represent the flow from the input streams to intermediate pools by proportion variables identifying the proportion of flow from input streams to intermediate pools. A third formulation, which is denoted as PQ-formulation, is developed by Tawarmalani and Sahinidis (2002). This formulation incorporates some valid inequalities on top of the Q-formulation. It should be noted that these constraints were derived by Quesada and Grossman (1995) for the applications in process networks. The equivalence of these three formulations is formally proved in Gupte *et al.* (2012). Other than those three; recently, Alfaki and Haugland (2012) proposed two new formulations denoted as TP-formulation and STP-formulation. TP-formulation uses a similar logic that Q-formulation does, but defines proportion variables regarding the proportion of the flow from intermediate pools to end products to replace the flow variables between intermediate pools and end products in P-formulation. It also includes the analogous valid inequalities that are used in PQ-formulation. Therefore it can be claimed that PQ-formulation and

TP-formulations are symmetric. Finally, STP-formulation combines PQ-formulation and TP-formulation by including both source proportion and end product proportion variables and corresponding constraints in the model.

In terms of number of variables, Q-formulation is smaller than P-formulation. However, for most of the cases, convex relaxations of P-formulation are tighter. On the other hand PQ-formulation, by the addition of the valid inequalities, provides a smaller problem with a tight relaxation (Misener and Floudas, 2009). Gupte *et al.* (2012) provides the comparison on the sizes of these formulations and a discussion on the relaxations of the problem and their tightness, and conclude the relaxation of the PQ-formulation is tighter than both P-formulation and Q-formulation. Alfaki and Haugland (2012) empirically compared the TP-formulation to PQ-formulation and claimed that the formulations do not have equal strength but no formulation dominates the other one. They also proved that the STP-formulation is not weaker than TP-formulation or PQ-formulation in the sense of their convex relaxations.

In this study, since a solution method that requires no relaxation is used and the smallest formulation is preferred, the Q-formulation which is given as in the study of Misener and Floudas (2009) is used with a few modifications. This is the same model defined in Ben-Tal *et al.* (1994) except it contains lower bound constraints on input material availability, end product demand and product quality as well as hard upper bounds on the variables. Also in this thesis, raw material cost parameters are defined with respect to their destination nodes. The summary for definitions of sets, parameters and decision variables used in the formulation is as follows:

Table 2.1. Sets, parameters, and decision variables for Q-formulation.

Sets	
I	Set of input streams (raw materials)
L	Set of pools
J	Set of output streams (end products)
K	Set of quality attributes
T_X	Set of (i,l) pairs for which input to pool connection exists
T_Y	Set of (l,j) pairs for which pool to output connection exists
T_Z	Set of (i,j) pairs for which input to output connection exists
Parameters	
c_{il}	Unit cost of raw material i sent to pool l
c_{ij}	Unit cost of raw material i sent to output j
d_j	Unit revenue for product j
A_i^L	Minimum required usage of raw material i
A_i^U	Maximum availability of raw material i
S_l	Capacity of pool l
D_j^L	Minimum required production of end product j
D_j^U	Demand upper limit of end product j
C_{ik}	Level of quality attribute k in raw material i
P_{jk}^L	Lower limit of quality attribute k in end product j
P_{jk}^U	Upper limit of quality attribute k in end product j
Decision variables	
q_{il}	Proportion of flow from input stream i to pool l among all flows into pool l
y_{lj}	Flow from pool l to output j
z_{ij}	Flow from input stream i to output j

Then the Q-formulation of the pooling problem can be given as the BLP defined in Equations 2.1 $-$ 2.10. In this formulation the objective function (2.1) minimizes the negative profit. Constraints (2.2) provide upper and lower limits on the raw material usages. Constraints (2.3) define the intermediate pool capacities. Minimum and maximum production limits of the end products are given in constraints (2.4). Constraints (2.5) and (2.6) impose the upper and lower quality requirements for linear blending. Constraints (2.7) provide that the proportion of flows to a pool add up to exactly one. Finally, constraints (2.8), (2.9) and (2.10) define the hard bounds on variables to tighten the feasible space. It should be noted that the bilinearities in the model occur in the objective function (2.1), the raw material limitation constraints (2.2) and the quality limitation constraints (2.5) and (2.6). The P-formulation and the STP-formulation of pooling problem are provided in Appendix A and Appendix B, respectively.

Q-formulation:

$$\min_{\substack{q_{il},y_{lj},\\ z_{i,j}}} \sum_{\substack{(i,l)\in T_X\\ (l,j)\in T_Y}} c_{il} q_{il} y_{lj} - \sum_{(l,j)\in T_Y} d_j y_{lj} - \sum_{(i,j)\in T_Z} (d_j - c_{ij}) z_{ij} \tag{2.1}$$

s.t.

$$A_i^L \le \sum_{\substack{l:(i,l)\in T_X\\ (l,j)\in T_Y}} q_{il} y_{lj} + \sum_{j:(i,j)\in T_Z} z_{ij} \le A_i^U \qquad \forall i \tag{2.2}$$

$$\sum_{j:(l,j)\in T_Y} y_{lj} \le S_l \qquad \forall l \tag{2.3}$$

$$D_j^L \le \sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \le D_j^U \qquad \forall j \tag{2.4}$$

$$\sum_{\substack{l:(l,j)\in T_Y\\ i:(i,l)\in T_X}} C_{ik} q_{il} y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik} z_{ij} \le P_{jk}^U \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{2.5}$$

$$\sum_{\substack{l:(l,j)\in T_Y\\ i:(i,l)\in T_X}} C_{ik} q_{il} y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik} z_{ij} \ge P_{jk}^L \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{2.6}$$

$$\sum_{i:(i,l)\in T_X} q_{il} = 1 \qquad \forall l \tag{2.7}$$

$$0 \le q_{il} \le 1 \qquad \forall (i,l) \in T_X \tag{2.8}$$

$$0 \le y_{lj} \le \min \left\{ S_l, D_j^U, \sum_{i:(i,l)\in T_X} A_i^U \right\} \qquad \forall (l,j) \in T_Y \tag{2.9}$$

$$0 \le z_{ij} \le \min \left\{ A_i^U, D_j^U \right\} \qquad \forall (i,j) \in T_Z \tag{2.10}$$

2.1.2. Solution Methods

As mentioned before, the general bilinear programming problem is strongly NP-hard. Also, Alfaki and Haugland (2012) formally proved that the pooling problem is strongly NP-hard in particular, by a polynomial reduction from the *Maximum Independent Vertex Set Problem.*

Many exact and heuristic solution techniques are suggested for the pooling problem. One can refer to Misener and Floudas (2009), Gupte *et al.* (2012) and Alfaki (2012) for a review of the solution methods. Obviously, all techniques that are suitable for nonlinear programming problems can be used to solve pooling problem. Some of the methods that are used to solve the pooling problem are summarized as follows, separated as exact and heuristic methods. Foulds *et al.* (1992) suggested a branch and bound algorithm based on convex underestimations which are achieved by replacing bilinear occurrences by new variables and bounding them. Also, Audet *et al.* (2004) provided a branch and cut algorithm by improving this technique. Liberti and Pantelides (2006), Wicaksono and Karimi (2008) and Gounaris *et al.* (2009) also provide more advanced relaxation techniques that are incorporated in branch and bound scheme. Moreover, apart from branch and bound variants, there are lagrangian relaxation based techniques. Floudas and Aggarval (1990) and Visweswaran and Floudas (1990) suggested lagrangian relaxation based approaches. Also, Ben-Tal *et al.* (1994), Adhya *et al.* (1999) and Almutairi and Elhedhli (2009) used lagrangian relaxation techniques to provide lower bounds on their global optimization algorithms. Moreover, Frimannslund *et al.* (2010) suggested a method that is based on linear matrix inequality relaxations.

Since exact methods can provide solutions to small sized problems, heuristic methods are used to find good enough solutions in reasonable time. The earliest inexact solution method is the iterative approach suggested by Haverly (1978), which estimates and fixes the quality attributes in pools and solves the remaining linear program. The new quality attributes are calculated by the flow values obtained by the results of the previous linear program and the procedure repeats until convergence occurs. Lasdon *et al.* (1979) used Successive Linear Programming (SLP) to solve pooling problem, which linearizes the problem using first order Taylor expansion at a point and solves the remaining LP, and repeats until convergence.

Audet *et al.* (2004) proposed a more general iterative heuristic named alternate heuristic (ALT), which uses the pooling problem's bilinear nature, becoming linear when one of the variable sets is fixed. The procedure starts with an initial feasible variable vector and solves the remaining LP by fixing this vector. Then the output is fixed on the next iteration and the procedure iterates until improvement stops. ALT converges to a local optimum if the solutions of the LPs are unique at each iteration of the algorithm.

Audet *et al.* (2004) also suggested a variable neighborhood search (VNS) which uses ALT as the local improvement method and a neighboring scheme that is suitable for the pooling problem. In this procedure, neighboring solutions are generated by fixing one of the variable sets that cause bilinearity and setting a number of basic variables as nonbasic in the remaining LP. The number of basic variables that will be forced to be nonbasic is given by the neighborhood size and at each iteration the variable that is fixed (the search subspace) is altered.

Recently, Alfaki (2012) proposed a greedy construction heuristic (GCH) that begins with choosing the most profitable end product node by considering only the raw materials and intermediate pools that can reach to that end product node, which corresponds to a linear program. Then for the next best end product, the subproblem that contains the pools and the input streams that can reach to that node is constructed and solved while preserving the previously allocated flows. The algorithm proceeds by augmenting the flows if they are profitable and terminates after all end product nodes are explored. In this method, although bilinearity exist in the subproblems that are solved in all of the iterations except the first, the number of bilinear terms are small and this may result in short run times for the solution of these subproblems by using general purpose solvers or appropriate methods.

```
Set $i := 1$
repeat
    Fix $q^i$ (or $y^i$) and solve the resulting LP, retrieve $y^i$ (or $q^i$)
    Fix $y^i$ (or $q^i$) and solve the resulting LP, retrieve $q^{i+1}$ (or $y^{i+1}$)
    $i := i + 1$
until stability is reached
```

Figure 2.1. Pseudo code of ALT.

```
Find an initial feasible solution $S$
Determine stopping condition maxIter, maximum neighborhood size $k_{max}$
Set $i := 1$ and subspace $\delta$
repeat
    $nSize := 1$
    repeat
        Generate $S'$ using $k$ and $\delta$
        $\delta := \overline{\delta}$
        Generate $S''$ using ALT with $S'$ as initial solution
        if $S''$ better than $S$
            $S := S''$
            $nSize := 1$
        else
            $nSize := nSize + 1$
    until $nSize > k_{max}$
    $i := i + 1$
until $i > maxIter$
```

Figure 2.2. Pseudo code of VNS.

```
Solve subproblem $P_0^j$ $\forall j$ and sort in increasing order
Initialize $S$ as flow vector containing the flows of the best $P_0$
Set $i := 1$
repeat
    Solve $i^{th}$subproblem $P_i$
    if  $P_i$ is profitable
        Augment flows of $P_i$ into $S$
    $i := i + 1$
until $i \geq |J|$
```

Figure 2.3. Pseudo code of GCH.

Pseudo codes of ALT, VNS and GCH are provided in Figures 2.1, 2.2, and 2.3 for pooling problem using the notation given in Section 2.1.1, since they will be used as benchmark methods.

2.2. Background on Particle Swarm Optimization

2.2.1. Overview of Particle Swarm Optimization

An evolutionary optimization technique for nonlinear functions, particle swarm optimization concept is first introduced by Kennedy and Eberhart (1995). The method is discovered through the study of movements of organisms in a bird flock or fish school. It uses a very simple concept that can be implemented easily and with little memory requirements. Basically, a set of particles are initialized randomly in the search space. Each particle moves in the space by keeping track of coordinates of its own best position encountered so far, *particleBest*, and the best position achieved by all of the particles in the population so far, *globBest*, with respect to their fitness values. The position change is determined by the velocity of the particles. Velocity calculation and position update mechanisms are given in Equations 2.11 and 2.12 respectively, in the notation used in this study (Eberhart and Kennedy, 1995). In Equations 2.11 and 2.12, V and

P stands for velocity and position vectors of a particle, and ϕ_1 and ϕ_2 are for two positive constants. $U(0, x)$ specifies a uniform random vector between 0 and x.

$$V_{i+1} = V_i + U(0, \phi_1)(particleBest - P_i) + U(0, \phi_2)(globBest - P_i) \tag{2.11}$$

$$P_{i+1} = P_i + V_{i+1} \tag{2.12}$$

PSO concept has links to both genetic algorithms and evolutionary programming. As in evolutionary programming, PSO depends on stochastic processes. Movement towards the personal and global best positions in PSO can be seen similar to the crossover operation in genetic algorithms. Like all evolutionary computation paradigms, fitness notion appears in PSO (Eberhart and Shi, 1998; Kennedy and Eberhart, 1995).

The basic PSO has a few number of parameters to determine beforehand. Population size is usually selected by considering problem difficulty and dimensions. Usually values between 20 and 50 are used as the number of particles. Also, $\phi_1 = \phi_2 = 2$ is quite common in the studies (Poli *et al.*, 2007). Furthermore, the study of Clerc and Kennedy (2002) provide a theoretical analysis about movements of particles in the search space and their convergence analytically. Zheng *et al.* (2003) and Trelea (2003) provide suggestions on parameter selection along with analytical convergence analysis. Shi and Eberhart (1998a), Shi and Eberhart (1999), Carlisle and Dozier (2001), Zhang *et al.* (2005) make suggestions on parameter selection provided by some empirical studies, although in general it is mentioned that the parameters may be problem specific.

Standard PSO algorithm does not consider constrained optimization problems. To deal with constrained problems, Parsopoulos and Vrahatis (2002) used penalty functions. In this approach maybe the most important factor is the penalty value since high penalty values can cause the algorithms to get trapped in local optima and low penalty values can result in problems in having feasible solutions. Hu and Eberhart

(2002) on the other hand, proposed a method that is based on maintaining feasibility. They initialize all particles in the feasible space and let the particles search the whole space. However, they let particles to update their best positions only if they are feasible. In this structure, it can be said that the biggest challenge is the equality constraints. Pulido and Coello (2004) suggested a different constraint handling mechanism that alters the selection of global best particle. This is done by a small change in fitness functions so that when comparing two particles, the feasible one is selected and if both particles are infeasible the one with the smallest infeasibility is selected.

The motivation to use PSO in this thesis is that it naturally handles continuous variables and performs well. However, variants of PSO to operate on discrete space are proposed as well. For such a variation, one can refer to Kennedy and Eberhart (1997) among others.

One can refer to Eberhart and Kennedy (1995) for the algorithmic steps of the original PSO. Some of the several extensions that were suggested on top this standard PSO are discussed in the next section. The papers of Schutte and Groenwold (2005) and Poli *et al.* (2007) should be referred for a more detailed survey on such extensions since in this study only the most important aspects are discussed.

2.2.2. Extensions on Original PSO

When updating the velocity of a particle, using its velocity in the former iteration provides the particle with the ability to explore search space and the procedure gains global search property (Shi and Eberhart, 1998b). However, leaving particle speed unattended can be harmful to the process, and the balance between global and local exploration abilities should be controlled. Mainly, this can be done by using bounds on velocities, inertia weight and constriction coefficients.

To prevent excessively large steps, velocity of a particle can be restricted to a range $[-V_{max},+V_{max}]$ with a proper choice of V_{max}. By setting V_{max} too small, global search ability is restricted and process will approach to local search. On the contrary, for the opposite case intensification may be compromised (Shi and Eberhart, 1998a).

Shi and Eberhart (1998b) proposed inertia weight concept which is basically a multiplier of the first term, previous velocity, in the velocity update Equation 2.11. Inertia weight can be utilized with or without V_{max} and can be a constant value or a time dependent function. They asserted that using inertia weight improves performance of the PSO. Also, it is stated that instead of using a fixed value, a linearly decreasing inertia weight increases the quality of the solution. Velocity calculation for the case where inertia weight is utilized can be seen in Equation 3.2 in Section 3.1.4, where inertia weight is denoted as ω.

Apart from constant and linearly decreasing inertia weight suggestions, Zheng *et al.* (2003) proposed linearly increasing inertia weight along with particle trajectory analysis. Chatterjee and Siarry (2006) studied nonlinearly decreasing inertia weight. Eberhart and Shi (2001) introduced a random component in inertia weight. Kentzoglanakis and Poole (2009) proposed oscillating inertia weight by utilizing a cosine term. Also, Yang *et al.* (2007), Jiao *et al.* (2008) and Alfi (2011) proposed dynamic adaptation for inertia weight value.

Clerc (1999) proposed the addition of a *constriction coefficient* may be useful to ensure convergence of the particles. Eberhart and Shi (2000) further explored the extension empirically and Clerc and Kennedy (2002) studied constriction coefficient analytically. Clerc and Kennedy (2002) commented that constriction coefficient can be implemented in many ways. One of the simplest methods of velocity update by using it is given in Equation 2.13, where Equation 2.14 and Equation 2.15 hold and χ is the constriction coefficient. When this constriction method is used, ϕ is usually set to 4.1 and $\phi_1 = \phi_2$ (Poli *et al.*, 2007).

$$V_{i+1} = \chi \left(V_i + U\left(0, \phi_1\right)\left(particleBest - P_i\right) + U\left(0, \phi_2\right)\left(globBest - P_i\right)\right) \tag{2.13}$$

$$\phi = \phi_1 + \phi_2 > 4 \tag{2.14}$$

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{2.15}$$

It should also be noted that PSO with constriction coefficient is algebraically equivalent to PSO with inertia weight. The mapping can be done be setting ω equal to χ and multiplying ϕ values with χ (Poli *et al.*, 2007).

Mendes *et al.* (2004) suggested a revision on the way a particle interact with other particles. Unlike the previous cases where only particle's self best and population best particle's information are used and the information from the remaining particles are ignored, they suggested *fully informed particle swarm* where each particle is affected by all of its neighbors. Velocity update of fully informed particle swarm can be given as in the Equation 2.16, where N_p is the number of neighbors that particle p has, and $nbr(n)$ is the n^{th} neighbor of particle p. It should be noted that if all particles only has itself and population best particle in its neighborhood, then this formulation is identical to standard version (Kennedy and Mendes, 2003).

$$V_{p,i+1} = \chi \left(V_{p,i} + \sum_{n=1}^{N_p} \frac{U\left(0,\phi\right)\left(P_{nbr(n),i} - P_{p,i}\right)}{N_p} \right) \tag{2.16}$$

Another discussion on PSO can be made about population topology. Population topology determines the set of particles that will affect the motion of a particle. Basically, population topologies can be characterized into two: static and dynamic topologies. In static topology neighborhoods remain constant during a run, on the other hand in dynamic topology neighborhoods can be updated. Static topology can be divided into two: global best topology, *gbest*, and local best topology, *lbest*. Global best topology can be conceptualized as a fully connected graph, therefore a particle's neighborhood is the whole population and the best particle in the population influences the particles (Poli *et al.*, 2007). Local best topology is introduced by Eberhart and

Kennedy (1995) in which neighborhood of a particle is defined as the nearest K many particles in the population array, where K is the neighborhood size. They concluded that *gbest* converges faster but there is a higher chance to be caught in local optima than *lbets* topology. Kennedy and Mendes (2002) and Kennedy and Mendes (2003) studied on numerous aspects of static network topologies and evaluated performances of them with respect to different algorithm versions. Mendes *et al.* (2004) proposed their fully informed particle swarm after a discussion on population topologies.

Suganthan (1999) suggested a dynamic topological structure by using the idea that *lbest* topology is better in exploring the search space and *gbest* is faster in convergence. Therefore they proposed to start the iterations with *lbest* and by dynamically increasing the neighborhood reaching *gbest* topology at the end of the run. They also defined the neighborhood in two ways; the simple way is considering the particles that are closest in the population array and the alternative way is constructing neighborhoods by considering the distances of particles in the search space. Also, Peram *et al.* (2003) and Liang and Suganthan (2005) studied on dynamic topologies.

2.3. Background on Simulated Annealing

A stochastic optimization method, simulated annealing is introduced by Kirkpatrick *et al.* (1983) and C̃erný (1985), independently. The technique is originally based on the connection between the statistical mechanics, which studies the behavior of systems in thermal equilibrium with high degrees of freedom, and combinatorial or multivariate optimization. SA metaheuristic is analogous to the physical annealing process of solids and named as such due to this analogy. The method can be classified as a random search technique but the key difference is its ability to escape being trapped at a local optimum by occasionally permitting worsening moves with a finite probability (Henderson *et al.*, 2003; Zomaya and Kazman, 2010).

In most general terms, the procedure is initialized at a temperature T, and it is decreased slowly by a cooling scheme. At each temperature level, a neighboring solution S' is generated from the current solution S by using predefined move functions. If the objective value of the neighboring solution, $F(S')$, is an improved value over the current objective value $F(S)$, then the move is always accepted. Moreover, the worsening moves are accepted with the probability function given in the Equation 2.17, for the minimization problem case. Therefore at high temperatures probability of accepting a worsening move is more than the lower temperature case, and this results in diversification at the beginning where the temperature is high and intensification at the end of the procedure. Also, new neighbors can be generated for a number of cycles at a temperature.

$$P(acceptance) = \exp\left(\frac{F(S) - F(S')}{T}\right) \tag{2.17}$$

During the course of the SA procedure, a neighboring solution is generated by performing a move on the current solution and it is independent of the previous solutions, therefore the algorithm can be interpreted as a series of Markov chains. By using this property and the fact that a sufficient number of iterations at a given temperature result in a stationary state distribution for an irreducible Markov chain; asymptotic convergence to global optimum is guaranteed for SA (Aarts *et al.*, 2005; Henderson *et al.*, 2003; Zomaya and Kazman, 2010).

The original SA algorithm basically requires a cooling schedule, a stopping criterion and a neighborhood definition to perform. Romeo and Sangiovanni-Vincentelli (1991) note that an effective cooling schedule is vital in reducing the time required to find a good solution. Cardoso *et al.* (1994) state that a fast cooling scheme can lead the algorithm end up at a poor local optimum and a slow cooling schedule result in very long computation time. Many of the cooling schedules in the literature are heuristics and can be classified in two as static and dynamic schedules. In static cooling, the parameters are fixed beforehand whereas dynamic schedules adjust the rate of decrease in temperature using the information gained during the algorithm execution. Most common cooling scheme which is an example of static cooling, geometric schedule is given as the Equation 2.18. In the equation, T_i states the temperature level at iteration i and α is the cooling coefficient which typically take values between 0.8 and 0.99 (Aarts *et al.*, 2005). The stopping of SA can be attained by setting a termination temperature or by finishing the execution after a number of consecutive decrease in temperature without having any accepted solutions (Zomaya and Kazman, 2010). Neighborhood definition is a problem specific choice for the SA algorithm as well as the size of the neighborhood. A large neighborhood size can lead to random sampling in a large portion of the feasible space whereas small neighborhood size can make it difficult to find good solutions in a reasonable time (Henderson *et al.*, 2003).

$$T_{i+1} = \alpha T_i \tag{2.18}$$

Although the SA algorithm originally worked on combinatorial optimization problems, it is applied to continuous optimization problems by mainly altering the neighboring strategies. For a few examples among them, Bohachevsky *et al.* (1986) proposed generating a uniform direction vector in the feasible region and moving along that direction for a predetermined step size. Corana *et al.* (1987) chose coordinates one by one as direction vectors in each iteration, and determined the step size randomly depending on the direction. Dekkers and Aarts (1991), Romeijn and Smith (1994) and Ali *et al.* (2002) proposed methods that used Markov chain approach for continuous optimization problems.

One can refer to Dowsland (1993), Henderson *et al.* (2003), Aarts *et al.* (2005) and Zomaya and Kazman (2010) for the formal algorithmic steps of the original SA, convergence analysis of the algorithm and a more detailed survey.

3. PROPOSED SOLUTION METHODS

Since Pooling problem is strongly NP-hard (Alfaki and Haugland , 2012; Audet *et al.*, 2004), two solution methods that are based on PSO and SA, which take advantage of the bilinear nature of the problem, are proposed to solve large instances effectively.

3.1. Particle Swarm Optimization Based Solution Method

3.1.1. Overview of the PSO-based Method

In simplest terms, the idea used in the PSO-based method is to choose one of the variable sets that cause bilinearity as the search variable and perform PSO operations on that variable set while fitnesses are calculated by solving the remaining LP.

As stated before a BLP reduces to an LP when one of the variable sets causing bilinearity is fixed, which can be solved easily and quickly. Therefore by fixing the values of the search variables in the model, fitnesses of particles and values of the rest of the variables can be calculated by solving the remaining LP. Also, by this way the constraints that are not entirely composed of the search variable set becomes irrelevant for the PSO methodology. If there exists a constraint that contains only the search variable and constants, the feasibility of that constraint is checked before fixing the variable and if needed, the position vector is repaired by using proper methods to make that constraint feasible. The search variable of the proposed solution method for pooling problem is selected as the q_{il} variable since it is bounded between zero and one, and also it is easy to repair the infeasibilities by normalizing the variables so that the constraint given by Equation 2.7 holds. Also, local improvement heuristic ALT, which is proposed by Audet *et al.* (2004), is performed occasionally during iterations to improve the fitnesses of particles in the population.

As explained in Section 2.2 there are several techniques exist to avoid getting stuck at local optima and increase the solution quality. In this thesis, *gbest* static topology, and a nonlinear inertia weight mechanism with a modification are used. The aim of the proposed procedure is to find a good set of values for the search variables, thus finding a good solution to the problem by using PSO framework.

The following subsections are dedicated to the details of the proposed PSO-based metaheuristic procedure. They include information about parameters and initialization of the population as well as the modifications to improve the solution quality and run time. Before presenting the details, general steps are given in the Figure 3.1.

```
Initialize parameters, set $i := 1$
Initialize position vector $P$ and velocity $V$ for each particle in the population
repeat
    for each particle in population
        Repair infeasibilities
        Calculate fitness F(P)
        if $F(P) < F(particleBest)$
            $particleBest := P$ and $F(particleBest) := F(P)$
            if $F(P) < F(globBest)$
                $globBest := P$ and $F(globBest) := F(P)$
    for each particle in population
        Update $V$ and $P$
    if Local improvement criteria is met
        Perform ALT
    if Population reduction criteria is met
        Remove the worst half of the population
    if Early termination criterion is met
        $i := maxIter$
    Update $\omega$ and $\phi_1$, $i := i + 1$
until $i > maxIter$
```

Figure 3.1. Pseudo code of PSO-based method.

3.1.2. Parameters

The original PSO needs a little number of parameters to perform, and for the proposed procedure there are a few additional parameters to determine as well as the PSO parameters.

First parameter is the population size which is usually based on dimension and the difficulty of the problem; and usually values between 20-50 are used (Poli *et al.*, 2007). In this study, this parameter is used as the initial population size and the number of particles is reduced during iterations as discussed in Section 3.1.3. Obviously the trade off between run time and solution quality affects the selection of the initial population size; since as the number of particles increases, the number of updates on the values of the fixed variables and the number of LPs solved at each iteration increases.

The parameters ϕ_1 and ϕ_2 provide upper bounds on the amount of random movement towards the personal best and population best solution vectors, respectively. One can assert that if ϕ_1 and ϕ_2 has small values, intensification will be slower. On the other hand, for the high values of ϕ_1 and ϕ_2, particles can begin to move in an uncontrolled manner. In this study the value of ϕ_2 is set independently and the value of ϕ_1 is initialized depending on ϕ_2 and updated at each iteration as discussed in Section 3.1.5.

Inertia weight ω can be seen as a term to restrict the velocity of a particle. For the values where ω is greater than 1, swarm will be unstable. For the values less than 1, diversification is stronger for the higher values of ω and intensification is anticipated for smaller values (Poli *et al.*, 2007). In this study, initial omega, final omega and omega curvature parameters are needed to be set and these parameters, as well as the update of ω during iterations is explained in Section 3.1.4.

Maximum number of iterations, $maxIter$, is the parameter that determines the termination of the algorithm, unless the procedure terminates prematurely due to early termination criteria. It is selected depending on the sizes of the problems in this study.

In general, parameter selection in this thesis is done by testing different values systematically on a set of problems and the details of this parameter setting phase is explained in Section 4.1.1.

3.1.3. Reducing Population Size

As the number of particles increases in the population the solution quality is expected to improve and as it decreases the time spent for optimization will reduce. Since we need to update and solve LPs for each particle at each iteration during the optimization procedure, which may be time consuming, a modification that compromises between solution quality and time is included in the procedure. We start by having a large population and reduce its size by removing particles during iterations. In this way we can increase the possibility of visiting promising areas in the solution space at the beginning and keep searching over the good particles and intensify to local optima at the later iterations. This reduction structure resembles the elitist strategy used in genetic algorithms. Eberhart and Shi (1998) argue that an explicit implementation of elitist strategy would be by carrying good particles to other iterations and eliminating a particle, probably the particle with the lowest fitness value, from the population.

In this thesis we implemented the idea in a way that particle elimination occurs at specified iterations. While the swarm size selected as a moderate value for most of the search, the exploration capability is tried to be increased by using larger swarm size during the early phases of the search .For this purpose, removing the half of the population twice, first at an iteration between $(3\% - 5\%)$ of $maxIter$ and secondly at an iteration between $(10\% - 15\%)$ of $maxIter$, is considered. At those points of the search, first ALT procedure is conducted for all of the particles and they are sorted with respect to their fitness values. Then the worst half of the particles is removed from the population. Initial size of the population is selected differently depending on the problem sizes, which are specified in Section 4.1.1, under numerical studies.

3.1.4. Inertia Weight Update

As mentioned in Section 2.2.2, instead of using bounds on velocities of particles, Shi and Eberhart (1998b) proposed inertia weight in order to restrict the velocities of the particles in the population which has a role in the exploration and local search trade off. A higher value for inertia weight results in larger differences in velocity in each iteration whereas small inertia weight means smaller updates in velocity. Thus, large values of inertia weight helps to explore new areas in the search space and small values means fine tuning around the current position. They asserted that using inertia weight improves performance and instead of using a fixed value, a linearly decreasing inertia weight increases the quality of the solution. Chatterjee and Siarry (2006) introduce a nonlinear variation in inertia weight. They used the formula in Equation 3.1 to update inertia weight ω, where $\omega curv$ specifies the nonlinear modulation index. For $\omega curv = 1$, the equation behaves as a linear adaptation, for $\omega curv > 1$, the rate in reduction of ω decreases as the number of iterations increase and the opposite is valid for $\omega curv < 1$. Equation 3.2 specifies the velocity update using inertia weight, where V is velocity, P_i is current position vector of the particle and $particleBest$ and $globBest$ are personal best position vector of the particle and position vector of global best encountered so far, respectively.

$$\omega_i = \left(\frac{maxIter - i}{maxIter}\right)^{\omega curv} (\omega_{initial} - \omega_{final}) + \omega_{final} \tag{3.1}$$

$$V_{i+1} = \omega_i V_i + U\left(0, \phi_1\right)\left(particleBest - P_i\right) + U\left(0, \phi_2\right)\left(globBest - P_i\right) \tag{3.2}$$

Chatterjee and Siarry (2006) tried several values for the parameters and for the nonlinear modulation index. They reported the best results for $\omega curv = 1.2$, which facilitates high enough values of ω during the first iterations to search the solution space better and low enough values to avoid large oscillations.

In this study, we update inertia weight using Equation 3.1 and $\omega curv = 1.2$ is selected as suggested at the work of Chatterjee and Siarry (2006) and $\omega_{final} = 0.2$ is used. However, a modification is done on the inertia weight mechanism. After examining the improvement pattern in a subset of problems, the nonlinear function is divided into three discrete segments. The parameters are initialized and at the end of the completion of 15% of $maxIter$ current omega value is reduced to a value less than one. Similarly, at the end of the completion of 75% of $maxIter$, current omega value is increased to a high value and at the rest of the iterations inertia weight reduction works the same; it reduces to its final value nonlinearly with $\omega curv = 1.2$. Figure 3.2 illustrates the inertia weight update mechanism where vertical axis represents the ω value at an iteration, horizontal axis represents iteration percentage and the dotted line represents the change in ω without the modification.

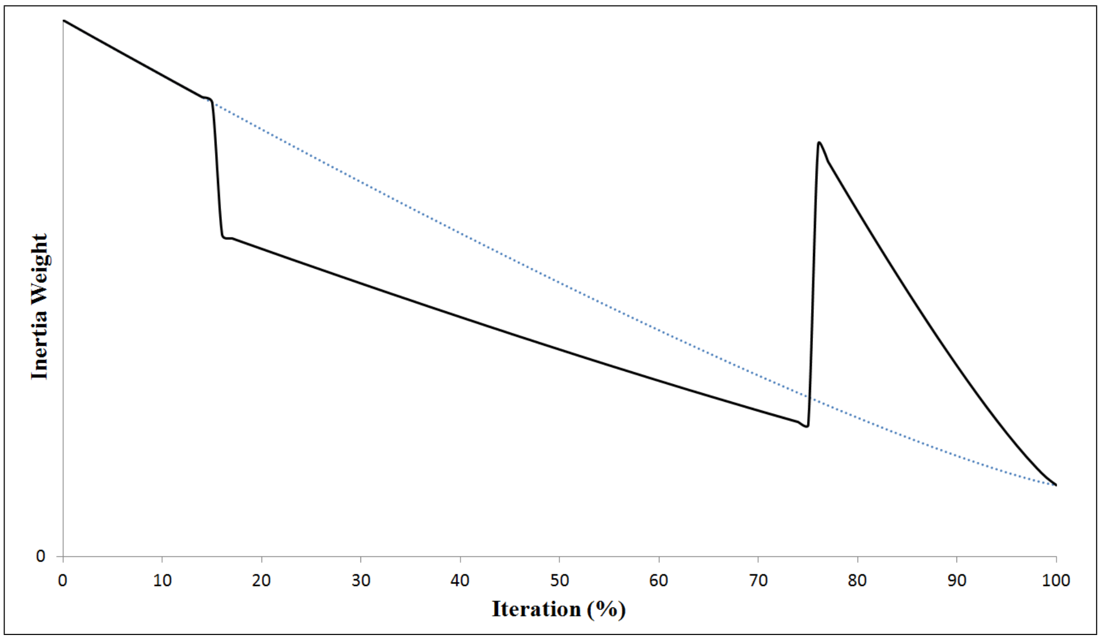

Figure 3.2. Inertia weight update mechanism.

This modification can be explained as such: The procedure starts with a high number of particles and the ALT procedure lets the particles to start their movements at their local optima. Also, we want the particles to explore promising areas before the population size is reduced. Therefore, at the early stages, global exploration is much more important and to do this, $\omega_{initial}$ is set to a high value. When 15% of the $maxIter$ is reached, where the population size is at its final value, inertia weight is reduced to a value less than 1, in this way we expect to observe convergent behavior. Because of performing ALT procedure occasionally and reducing inertia weight, the frequency of improvements in the population reduces as the iterations move on. Therefore, it is aimed that the particles explore their neighborhood a bit further by increasing inertia weight at the iteration where 75% of the *maxIter* is performed.

3.1.5. ϕ_1 Update

The ϕ_1 parameter affects the random movement towards the personal best position of the particle and ϕ_2 parameter affects movement towards population best position. Therefore, it can be stated that ϕ_1 helps us to search the neighborhood of the particle and ϕ_2 provides intensification towards population best. By using these properties, it is decided to keep the value of ϕ_2 fixed throughout the iterations but gradually reduce the value of ϕ_1 to a certain level by beginning at a higher value than ϕ_2, in order to provide the relative difference. In this way, at the first iterations particles will move around their neighborhoods more than they intensify, thus diversification will be dominant. And as the iteration limit comes closer, particles' movement towards population best will be dominant and intensification will be provided. Equation 3.3 states the update mechanism, where ϕ_1 reaches to the half of ϕ_2 at termination. $\phi_{1,initial}$ is set as a multiple of ϕ_2 and initial values of these are provided under numerical studies at Section 4.1.1.

$$\phi_{1,i} = \left(\phi_{1,initial} - \frac{\phi_2}{2}\right)\left(\frac{maxIter - i}{maxIter}\right) + \frac{\phi_2}{2} \tag{3.3}$$

3.1.6. Local Improvement

As described in Section 2.1.2, ALT is a natural solution method for BLPs and provides local optimum in finite number of steps given that each LP has a unique solution (Audet *et al.*, 2004). In order to take advantage of that, at the first and at every 50 iterations ALT procedure is performed as well as at the termination. Moreover, to maintain a good *globBest*, best three particles in the population are subjected to

ALT at every 10 iterations.

3.1.7. Early Termination

Since solution time is an issue, an early termination rule is proposed to decide on whether or not to terminate the procedure before completing $maxIter$. It is based on observations on the improvement patterns of a subset of problems during preliminary experiments. The pseudo code of the rule is given in Figure 3.3. With this rule, the procedure cannot be terminated before 66% of the iterations are completed, in order to avoid premature termination.

Usage of early termination can be decided depending on the convergence of the procedure, by observing the improvement patterns. In this study, the decision to use early termination rule is based on the problem sets at hand.

```
if 80% of maxIter completed and no improvement for last 5% of maxIter
    return true
else if 71% of maxIter completed and no improvement for last 10% of maxIter
    return true
else if 66% of maxIter completed and no improvement for last 30% of maxIter
    return true
return false
```

Figure 3.3. Pseudo code of early termination rule.

3.1.8. Population Initialization

In order to begin with the procedure, the particles in the population must have initial position and velocity vectors. In this study, velocity vectors of all particles are selected as zero and they are expected to initialize themselves. On the other hand, initial positions can be assigned to particles either randomly, or by using some heuristics, or by a combination of heuristic and random positions. In this thesis, the position vectors of the particles are initialized as suggested in Audet *et al.* (2004); each variable is set to zero with a 0.5 probability and if it is decided to be nonzero, it is initialized randomly between $(0, 1)$ range.

3.2. Simulated Annealing Based Solution Method

3.2.1. Overview of the SA-based Method

The proposed SA-based method basically applies the original SA, but uses the neighborhood definition that is proposed for the VNS procedure in Audet *et al.* (2004), which is explained in the Section 2.1.2. The SA procedure is further enhanced by starting with a large neighborhood size and then reducing it gradually during the course of the search.

To illustrate the neighbor generation for the pooling problem, let $S = (q', y', z')$ be the current solution. Then (q', z') is an extreme point of this solution in a subspace $\delta = (q, z)$ where $y = y'$ and this subspace is as a convex polytope associated with the LP defined by fixing the variable y. The neighborhood of size one of this extreme point in subspace δ is defined as the all extreme points that are reachable by taking one basic variable out of basis. Then in general terms, a neighbor $S' = (q'', y', z'')$ in a neighborhood of size k is generated by changing randomly k elements in the basis of the solution in the given subspace. In next neighbor generation, q variables will be fixed. In case S' is accepted, the search moves to this new solution and $q = q''$ is fixed on the next iteration; otherwise $q = q'$ is fixed. In this way, independent of whether the search has moved to a new solution, at each neighbor generation the subspace δ is altered to its complement. The complement of subspace (q, z) is defined as the subspace (y, z). Thus, solutions in (q, z) and (y, z) are generated interchangeably.

In the proposed SA-based method, neighborhood size is determined depending on the iteration. The initial and final neighborhood sizes are initialized at the beginning of the search and the current value is updated at each iteration. Also, the number of cycles that the search runs for each temperature level is dynamic and increases geometrically. The aim is to explore the feasible space with a larger neighborhood size at the beginning and intensify at the end of the procedure with longer cycles. The termination of the algorithm occurs when the iteration count reaches a parametrized maximum number of iterations limit.

The following subsections are devoted to the details of the proposed SA-based method; detailed information on parameters, initialization and the modifications to improve the solution quality are explained. Before presenting the details, general algorithmic steps are provided in the algorithm in Figure 3.4. In the pseudo code, S denotes current solution and S' represents its neighbor, and their objective values are given by $F(S)$ and $F(S')$, respectively. Parameter $nSize$ stands for current neighborhood size, $cLen$ for current cycle length and T for current temperature.

```
Initialize parameters, set $i := 1$ and subspace $\delta$
Generate initial solution $S$
repeat
    for cycle:=1 to cLen
        Generate Neighboring Solution $S'$ from $S$ using $nSize$ and $\delta$
        if $F(S') < F(S)$
            $S := S'$
        else if $Random(0,1) < exp(\frac{F(S)-F(S')}{T})$
            $S := S'$
        $\delta := \overline{\delta}$
    Update $T$, $nSize$, $cLen$
    $i := i + 1$
until $i > maxIter$
```

Figure 3.4. Pseudo code of SA-based method.

3.2.2. Parameters

In the implementation of the SA-based method, a few additional parameters are used in order to initialize and perform the procedure. The cooling scheme is selected as the geometric cooling schedule and the cooling coefficient parameter, α, is searched for different values in order to assess the run time and solution quality tradeoff. The geometric temperature update used as presented in Equation 2.18.

Since initial temperature has an important role in the solution quality, instead of assigning a constant value for the initial temperature T_0 for all problem instances, a meaningful value is determined as suggested by Dowsland (1993) for each instance at the beginning of the search. For this purpose, a sample solution points and a neighbor solution for each of them are generated. Then the average of individual absolute differences in fitness values of the solution pairs, $\overline{|\Delta f_0|}$, and the initial acceptance ratio, IAR, are used to find T_0 as in Equation 3.4 below. The value of IAR parameter is searched for different values to find a good value.

$$T_0 = \frac{\overline{|\Delta f_0|}}{\ln\left(\frac{1}{IAR}\right)} \tag{3.4}$$

In order to determine the termination criterion, the value of the maximum number of iterations, $maxIter$, is calculated such that at the end of the procedure temperature reaches to the desired level, T_f, under geometric cooling schedule. Therefore, Equation 3.5 can be used to calculate the $maxIter$ parameter. However for similar reasons, a generic calculation method is proposed instead of using a single T_f for all problem instances. Therefore $maxIter$ is calculated by using the final acceptance ratio parameter, FAR, as in Equation 3.6 which can be derived from Equation 3.5 and k stands for the $\overline{|\Delta f_0|}/\overline{|\Delta f_f|}$ ratio. In this study, by experimenting over a subset of instances k is estimated as 2. The value of FAR parameter is selected as 0.001 in this thesis.

$$maxIter = \log_\alpha\left(\frac{T_f}{T_0}\right) \tag{3.5}$$

$$maxIter = \log_{\alpha}\left(\frac{\ln(IAR)}{k\ln(FAR)}\right) \tag{3.6}$$

The neighborhood size parameter, $nSize$, that affects the new solution generation is adaptive and it decays geometrically as cooling schedule. In this way, having a larger neighborhood size helps diversification at the beginning and as the procedure reaches termination small steps in the neighborhood space leads to intensification. The final neighborhood size, FNS, is selected as 1 in this study for all instances. However, in order to take problem sizes into account initial neighborhood size, INS, is determined using initial neighborhood size coefficient, β. The calculation of INS is given in Equation 3.7. As mentioned before, neighbors are generated by fixing one of the variable sets that cause bilinearity and removing $nSize$ number of basic variables from the basis of the remaining LP as in the study of Audet *et al.* (2004). Since at each iteration one of those variable sets are perturbed, average number of variables that cause bilinearity is multiplied by the parameter β. Maximum of 5 and this value is selected in order to avoid having very small INS. Similar to some other parameters, the value of β is selected among different values to have a good performance. As a result of these calculations, neighborhood update coefficient, γ, is determined such that at the end of the procedure FNS is reached. Equation 3.8 illustrates the calculation of γ. The $nSize$ parameter is updated as in Equation 3.9.

$$INS = \max\left(5, \beta\,\frac{\textit{number of variables causing bilinearity}}{2}\right) \tag{3.7}$$

$$\gamma = \left(\frac{FNS}{INS}\right)^{1/maxIter} \tag{3.8}$$

$$nSize_{i+1} = \gamma\, nSize_i \tag{3.9}$$

The parameter that determines the number of cycles at each temperature level, $cLen$, increases geometrically in order to be able to explore close neighborhood when the procedure is close to termination. For this purpose, initial cycle length, ICL, and final cycle length, FCL, are used. ICL parameter is used as 1 for all cases whereas

FCL depends on problem instances. Using a similar idea as before, cycle length update parameter, θ, is calculated as in Equation 3.10 and $cLen$ is updated as in Equation 3.11.

$$\theta = \left(\frac{FCL}{ICL}\right)^{1/maxIter} \tag{3.10}$$

$$cLen_{i+1} = \theta\, cLen_i \tag{3.11}$$

3.2.3. Local Improvement

In order to improve the quality of the current solution, local optimization method ALT is used for the SA-based procedure as in the other method. During the procedure, at first iteration and at the termination local search by ALT is performed. Moreover, whenever a neighboring solution improves the current solution ALT is carried out. The reason is to reach the local optimum easily and to avoid spending time unnecessarily in a valley.

3.2.4. Solution Initialization

For the initial solution generation of the algorithm, the same method used in the PSO-based method is utilized. The position vector of the solution is initialized as proposed in Audet *et al.* (2004); each variable is set to zero with a 0.5 probability and if it is decided to be nonzero, it is initialized randomly between $(0, 1)$ range.

4. NUMERICAL RESULTS AND PERFORMANCE OF THE PROPOSED SOLUTION METHODS

4.1. Test Problems and Experimental Setting

There is a number of problem instances in the literature that are used widely in testing the performances of the solution techniques for the pooling problem (Adhya *et al.*, 1999; Audet *et al.*, 2004; Ben-Tal *et al.*, 1994; Foulds *et al.*, 1992; Haverly, 1978). However, these benchmark problems represent small to medium size problems (Misener and Floudas, 2009). Recently, Alfaki and Haugland (2012) presented randomly generated large instances of the pooling problem.

In this study, in order to study the performance of the proposed heuristics, mostly large problem instances are used, and generated when necessary. The instances that are used in experiments can be divided into four sets, roughly by problem sizes. Set 1 contains 14 problems: the third example given in Adhya *et al.* (1999), referred as AST3, the last randomly generated instance by Audet *et al.* (2004), referred as R19, and 12 other problems that are fabricated by combining different problems using AST3 and the last four problems of Audet *et al.* (2004). These are referred as R19AST3_1, R18R17_1, R18R16_1, R17R16_1, R19AST3_2, R18R17_2, R18R16_2, R17R16_2, R19AST3_3, R18R17_3, R18R16_3, R17R16_3. These 12 problems were generated to obtain medium to large sized problems before Alfaki and Haugland (2012) published their study. They are fabricated by combining two separate problems into one and the name of the problem consists of those two instances. The suffixes "_1", "_2" and "_3" determines the combination technique. For the instances with suffix "_1", the two problems are just combined by increasing the indices of the sets of the second problem by the cardinality of the relevant sets of the first problem. Therefore it can be said that these are two separable problems expressed as one. For the instances with suffix "_2", on top of the former problem, some additional network links and their relevant cost parameter values are generated randomly and included in the problem. Lastly, for the instances with suffix "_3", additional to the previous combination, quality attributes are defined randomly for some of the raw materials and end products which are connected to the other problem by the additional links. There are 20 problems in set 2, set 3 and set4. These instances are given in Alfaki and Haugland (2012) and set 2 includes group A, set 3 contains group B and set 4 is composed of group C problems provided in their study. All of the instances that are used in this study can be downloaded in a GAMS readable format from http://www.bufaim.boun.edu.tr/Pooling_Instances.zip.

The sizes of the problems for the Q-formulation are given in the Table 4.1 to provide a rough understanding about the problem difficulties. The first two columns of Table 4.1 state the set and the name of the problem and the succeeding columns identify the number of input streams (ni), intermediate pools (nl), end products (nj) and quality attributes (nk). Columns seven to ten identify the number of variables (vars), number of linear equality constraints (leq), number of linear inequality constraints (lin) and number of bilinear inequality constraints (bin).

To analyze the performances of the two proposed solution methods, the results are compared with heuristic solution techniques that are presented in the literature. Multi-start version of ALT (MALT), VNS presented in Audet *et al.* (2004) and the greedy construction heuristic (GCH) offered in Alfaki (2012) are used for comparison and the pseudo codes of those are given in Figures 2.1, 2.2 and 2.3, respectively. In the study of Audet *et al.* (2004), multi-start version of SLP method is dominated by MALT and VNS for all of their randomly generated test instances, therefore it is omitted as a solution method for comparison. Apart from those, an open source large scale nonlinear optimization software IPOPT (Wächter and Biegler, 2006), a commercial software package KNITRO which is a large scale mathematical optimization software specialized in nonlinear optimization (Byrd *et al.*, 2006) and a commercial nonconvex optimization software BARON (Tawarmalani and Sahinidis, 2005) are used as general purpose solvers for experimentation. BARON guarantees global optimality; however KNITRO and IPOPT are local optimizers. For the heuristic methods and general purpose solvers Q-formulation of the pooling problem is used. Moreover, in order to

Table 4.1. Sizes of the test instances.

	Problem	**ni**	**nl**	**nj**	**nk**	**vars**	**leq**	**lin**	**bin**
Set 1	**AST3**	8	3	4	6	20	3	55	64
	R19	12	10	11	4	182	10	382	112
	R19AST3_1	20	13	15	10	202	13	437	176
	R18R17_1	22	10	8	34	123	10	235	316
	R18R16_1	20	14	13	33	216	14	446	334
	R17R16_1	22	14	13	7	213	14	435	130
	R19AST3_2	20	13	15	10	235	13	504	340
	R18R17_2	22	10	8	34	157	10	297	588
	R18R16_2	20	14	13	33	253	14	507	898
	R17R16_2	22	14	13	7	253	14	519	226
	R19AST3_3	20	13	15	10	235	13	504	340
	R18R17_3	22	10	8	34	157	10	297	588
	R18R16_3	20	14	13	33	253	14	507	898
	R17R16_3	22	14	13	7	253	14	519	226
Set 2	**A0**	20	10	15	24	171	10	376	758
	A1	20	10	15	24	179	10	416	712
	A2	20	10	15	24	192	10	400	760
	A3	20	10	15	24	218	10	457	760
	A4	20	10	15	24	248	10	491	760
	A5	20	10	15	24	277	10	570	760
	A6	20	10	15	24	281	10	571	760
	A7	20	10	15	24	325	10	666	760
	A8	20	10	15	24	365	10	720	760
	A9	20	10	15	24	407	10	812	760
Set 3	**B0**	35	17	21	34	384	17	768	1498
	B1	35	17	21	34	515	17	965	1498
	B2	35	17	21	34	646	17	1248	1498
	B3	35	17	21	34	790	17	1464	1498
	B4	35	17	21	34	943	17	1779	1498
	B5	35	17	21	34	1044	17	1947	1498
Set 4	**C0**	60	30	40	40	811	30	1604	3320
	C1	60	30	40	40	1070	30	2101	3320
	C2	60	30	40	40	1278	30	2523	3320
	C3	60	30	40	40	1451	30	2802	3320

analyze the solution quality in depth and let the optimization packages use the advantage of a tighter relaxation, STP-formulation of pooling problem is also experimented as well as the Q-formulation for the general purpose solvers. The STP-formulation is given in the Appendix B.

In order to improve the solution quality and reduce the effect of the random seed that initializes the random number stream that is used during the procedure, the solutions are replicated with different number of seeds depending on their run times. The solutions for set 1 and set 2 are replicated five times and for set 3 three times. Therefore the best objective value and the sum of run times of the replications are reported. For set 4 however, the average objective values and run times of five replications are reported due to long run times.

The VNS procedure that is reported in Audet *et al.* (2004) needs two parameters: maximum neighborhood size and number of iterations which is the stopping condition of the algorithm. Maximum neighborhood size is selected as 100 as it is used in Audet *et al.* (2004). In order to provide a fair comparison, stopping condition is imposed by a time limit which is set as the time spent on the PSO-based solution method for that instance. The VNS procedure is replicated in the same way of the runs of the proposed solution methods.

The MALT procedure requires number of starting points as the parameter in the study of Audet *et al.* (2004). However, again for the sake of a fair comparison, the stopping condition is given by a time limit in this study. New position vectors are generated and subjected to ALT procedure until the time limit, which is determined by the time spent on the PSO-based method for the particular instance, is exceeded.

The only parameter GCH requires is the time limit for the procedure. Although one hour is used as time limit in Alfaki (2012), the time limits allowed here the same as all other methods for a fair comparison.

The proposed solution methods, VNS, MALT and GCH are coded in C# language via Microsoft Visual Studio 2010. In order to solve the LPs in PSO-based, SA-based, VNS and MALT procedures IBM ILOG CPLEX 11.0 is employed.

For the nonlinear optimization software, BARON, KNITRO and IPOPT, time limits are imposed as other heuristics and memory limits are set as 3000 Mbytes to avoid memory insufficiency. The maximum iteration limit for KNITRO and IPOPT are set to unlimited to avoid termination due to iteration limit before the time limit is reached. KNITRO has three options for the algorithm used for solution; two of which are interior point methods (Interior/Direct algorithm and Interior/CG algorithm) and the other one (Active Set algorithm) follows an exterior path. As default, KNITRO uses the first algorithm and switches to the second algorithm if needed during the run time, which use interior methods and terminates when the feasibility error is within its limits. However, it should be noted that the solution reported in this case still has infeasibility even though it is small. When feasibility tolerance is set to zero, KNITRO fails to find a feasible solution for any of the problem instances. In this thesis, Active Set Algorithm with multi-start option (ms_enable true, ms_terminate 1) is used to eliminate that problem. The rest of the options of the solvers are left as their default values.

The mathematical models are coded on GAMS IDE version 23.6.5; and BARON version 9.0.6, KNITRO version 7.0.0 and IPOPT version 3.8 are run through the GAMS interface to solve the models. Finally, all experiments are carried out on a PC with 3.07 GHz CPU and 8 GB RAM, running under 64-bit Windows 7 operating system.

In Chapter 3, where the proposed solution methods are presented, required parameters are described and values are specified if they are fixed for all problem instances. The other parameter values that depend on the instance sets are presented on the two following subsections for PSO-based and SA-based approaches, respectively.

4.1.1. PSO-based Method Parameter Values

PSO-based method requires initial population size, $maxIter$, early termination rule usage, $\omega_{initial}$, ϕ_2 and initial multiplier for ϕ_1 values other than the ones that are stated before. Since run time of the procedure is important and must be within reasonable limits, and initial population size and maximum number of iterations are the main factors affecting it; initial population sizes and $maxIter$ are determined by the size of the problems. By using the observations in the preliminary experiments, for the problems in the set 1 initial population size is set as 100. For the instances in the set 2, initial population size is selected as 60, and 40 is chosen for set 3 and set 4. The population size is reduced to its half at 5^{th} and 15^{th} iterations. Value of $maxIter$ is set as 100 for set 1 and set 2. Also, early termination rule is applied for those two sets. On the other hand, 150 is selected as $maxIter$ and early termination rule is not used for the rest, namely set 3 and set 4. For the modification in ω value, it is chosen that the omega value is reduced to 0.9 at the completion of %15 of $maxIter$, and increased to 1.2 at the completion of %75 of $maxIter$.

The values of $\omega_{initial}$, $\phi_{1,initial}$ and ϕ_2 are selected among different alternatives by experimenting over a subset of 8 problem instances, namely AST3, R19AST3_2, R17R16_3, A1, A4, A9, B0 and B4, in order to avoid over fitting. For $\omega_{initial}$, a high value (h) 1.5 and a low value (l) 1.2; for ϕ_2 1.25, 1.0 and 0.75 (h, m and l, respectively) are tried. $\phi_{1,initial}$ is initialized as a multiple of ϕ_2 and this multiplier is searched among two values, namely 1.5 and 2.0 (h and l, respectively). Full factorial design for the three parameters is used and each parameter setting is denoted in a three letter format where first letter states the level of $\phi_{1,initial}$ multiplier, the second represents ϕ_2 level and the last one states the $\omega_{initial}$ level. To check whether there is a significant difference among those 12 parameter settings Friedman test is used (Hollander and Wolfe, 1999). Friedman test is a nonparametric statistical test that is used to compare the medians of observations repeated on the same subjects. The null hypothesis is that all treatment effects are equal, and the alternative hypothesis is at least two of the treatment effects are not equal; where the parameter settings denote the treatment effects. It is the nonparametric alternative of repeated measures analysis of variance test. Since the normality assumption does not hold, the nonparametric Friedman test is applied in this study. Basically, for each instance, each parameter setting is assigned a rank such that small observations get a smaller rank. Then ranks of each parameter settings are summed and Friedman test statistic is calculated. The tests are conducted using statistical software package Minitab 16. Table 4.2 provides the Friedman test results below and it fails to reject the null hypothesis since $p > 0.05$. Although there is no parameter setting that is significantly different, the parameter setting with the lowest rank is selected, namely "hmh". All parameters used for PSO-based method are summarized in Table 4.3 with respect to the problem sets, where Est Median stands for estimated median.

Table 4.2. Friedman test result for PSO-based method parameter settings.

Setting	Observation Count	Est Median	Sum of Ranks
hhh	8	-23753	55.0
hhl	8	-23778	50.0
hlh	8	-23691	66.0
hll	8	-23707	59.5
hmh	8	-23865	26.5
hml	8	-23761	54.0
lhh	8	-23758	55.5
lhl	8	-23777	48.5
llh	8	-23804	42.0
lll	8	-23562	72.5
lmh	8	-23770	46.0
lml	8	-23744	48.5
p = 0,193 (adjusted for ties)			

Table 4.3. PSO-based method parameter summary.

	Set 1	**Set 2**	**Set 3**	**Set 4**
$popSize$	100	60	40	40
$maxIter$	100	100	150	150
$\phi_{1,initial}$	2.0	2.0	2.0	2.0
ϕ_2	1.0	1.0	1.0	1.0
$\omega_{initial}$	1.5	1.5	1.5	1.5
ω_{final}	0.2	0.2	0.2	0.2
$\omega curv$	1.2	1.2	1.2	1.2

4.1.2. SA-based Method Parameter Values

Apart from the previously reported parameter values, SA-based method requires initial neighborhood size coefficient, β, initial acceptance ratio, IAR, cooling coefficient, α and final cycle length, FCL.

A similar approach is used in the determination of parameters that is used in PSO-based method is adopted for the parameters of SA-based approach. Experimentations are carried out over the same subset of problems and full factorial design is used where the levels are denoted by letters for each parameter. For β, 0.2, 0.1 and 0.05 (h, m and l, respectively), for IAR, 0.99, 0.95 and 0.9 (h, m and l, respectively) and finally for α, 0.95 and 0.9 (h and l, respectively) are tried. Friedman test is used to check whether there is a significant difference among those 18 parameter settings and Table 4.4 presents the result of the test.

From Table 4.4, it can be seen that not all the parameter settings are equal since $p < 0.05$. Since the small rank values represent better performance, the two settings that provide the best performance, “hhh” and “hlh” are tested among themselves to see if there is a significant difference between them and Table 4.5 shows the test results. Since $p > 0.05$, it can be said that parameter settings “hhh” and “hlh” are statistically equal. As a result, for set 1 and set 2 instances "hhh" setting, for set 3 and set 4 problems "hlh" parameter settings are used, in order to reduce the time spent on calculations.

Table 4.4. Friedman test result for SA-based method parameter settings.

Setting	Observation Count	Est Median	Sum of Ranks
hhh	8	-19769	33.5
hhl	8	-19748	68.5
hmh	8	-19750	60.5
hml	8	-19717	92.5
hlh	8	-19757	45.5
hll	8	-19681	99.5
mhh	8	-19750	52.5
mhl	8	-19729	87.5
mmh	8	-19738	72.5
mml	8	-19713	98.5
mlh	8	-19723	78.5
mll	8	-19671	117.5
lhh	8	-19740	67.5
lhl	8	-19753	68.5
lmh	8	-19758	58.5
lml	8	-19714	94.5
llh	8	-19740	72.5
lll	8	-19710	99.5
p = 0,003 (adjusted for ties)			

Table 4.5. Friedman test result for SA-based method parameters for the best two settings.

Setting	Observation Count	Est Median	Sum of Ranks
hhh	8	-20443	10.5
hlh	8	-20442	13.5
p = 0,257 (adjusted for ties)			

After selection of these parameters, an important time difference between PSO-based and SA-based methods for the set 1 and set 2 problems is observed. Based on this observation, in order to increase the solution quality, α is increased to 0.97 and FCL is increased to 20 for set 1 from the default value 10. For the set 2 instances, FCL is increased to 15.

Although most of the time SA-based approach terminates earlier, all instances for the SA-based procedure are limited by time, which is determined by the time spent on the PSO-based method for the particular instance for a fair comparison.

All parameters used for SA-based method are summarized in Table 4.6 with respect to the problem sets.

Table 4.6. SA-based method parameter summary.

	Set 1	**Set 2**	**Set 3**	**Set 4**
IAR	0.99	0.99	0.9	0.9
FAR	0.001	0.001	0.001	0.001
β	0.2	0.2	0.2	0.2
FNS	1	1	1	1
α	0.97	0.95	0.95	0.95
ICL	1	1	1	1
FCL	20	15	10	10

4.2. Results and Comparison

The primary criterion in evaluation of the performances in this section is the objective values of the solution techniques. As mentioned before, all methods except PSO-based approach has implied upper limits on the run time and this limit is determined by the time spent on PSO-based method. However, there may be some excess in times since the iterations are allowed to be completed if time limit is reached. As previously stated, for VNS, PSO-based method and SA-based method, total time spent and minimum of the replications are presented except for set 4, where the average time and objective values are reported. In all of the tables that present the numerical values, if a solution technique fails to find a feasible solution within the given time limit or terminates due to convergence to an infeasible point, the objective value of that instant is left with a dash. For the results of BARON, if the optimal solution is found, then the lower bound value is left with a dash.

Before comparing the experimentation results with the proposed methods, the results of the Q-formulation and the STP-formulation are compared for each solver. For KNITRO, 12 of the instances have equal objective values, for 15 of the instances Q-formulation yielded lower results whereas opposite is true for 7 cases. For the runs with IPOPT, for 15 of the instances Q-formulation and for 10 of the instances STP-formulation turned out to be better. Also, it can be said that for most of the cases, run times are worsened when switching to STP-formulation using IPOPT solver. When the upper bound values of BARON are compared, it is seen that STP-formulation resulted in lower objective values for 8 of the cases, 2 of them belonging to the set 1 of instances, and Q-formulation provided better results for 12 cases, all of which belonging to the other sets. However, as it was expected, the lower bounds of Q-formulation are inferior to the ones of STP-formulation for all instances. It is also noteworthy that KNITRO and IPOPT failed to find feasible solutions for most of the cases in sets 2, 3 and 4. Tables 4.7, 4.8 and 4.9 provide the results of these comparisons where the better objective values are presented in bold.

The comparison between the performances of PSO-based method and the general purpose solvers are provided in the Table 4.10 for set 1 and the other sets are given in the Table 4.11. In this comparison, each of the solver solutions are compared to the proposed PSO-based method in a pair-wise manner. If an objective value is strictly better than the PSO-based method then it is presented in bold.

When the Q-formulation solved with KNITRO and PSO-based methods are compared, it can be seen that in 29 instances in all 34, PSO-based method resulted in better

Table 4.7. Q-formulation and STP-formulation comparison for KNITRO.

		KNITRO Q		**KNITRO STP**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**
Set 1	**AST3**	**-559.62**	0.1	-65.00	0.1
	R19	-4524.18	0.2	-4524.18	94.5
	R19AST3_1	-5083.80	47.6	-5083.80	2.8
	R18R17_1	**-1981.82**	0.6	-1429.16	0.2
	R18R16_1	-2841.67	0.4	**-2858.50**	0.4
	R17R16_1	**-3462.00**	0.3	-3322.00	0.3
	R19AST3_2	-4594.18	1.5	-4594.18	1.0
	R18R17_2	**-2034.16**	0.6	-1429.16	0.7
	R18R16_2	-2841.67	0.8	-2841.67	8.1
	R17R16_2	**-3506.79**	1.6	-3322.00	0.3
	R19AST3_3	-3779.50	3.6	**-4078.44**	1.8
	R18R17_3	**-830.32**	1.2	-91.51	0.1
	R18R16_3	**-1715.28**	3.3	-865.75	0.2
	R17R16_3	-2602.82	14.2	**-2624.86**	1.3
Set 2	**A0**	**-33838.65**	241.5	-27939.13	522.9
	A1	**-23577.65**	8.9	-20668.93	58.8
	A2	-	855.0	**-6443.82**	50.4
	A3	-33493.38	212.8	**-34844.11**	102.7
	A4	**-39656.81**	400.2	-	1093.2
	A5	**-26265.05**	1252.9	-	1191.0
	A6	**-41945.96**	515.9	-41906.40	1104.9
	A7	-	1486.0	-	1486.0
	A8	-	1309.0	-	1309.0
	A9	**-21593.58**	943.2	-20774.54	1249.5
Set 3	**B0**	**-37210.20**	91.6	-	2135.0
	B1	-57932.09	1292.8	**-58299.34**	2465.6
	B2	-	3624.0	-	3624.6
	B3	-	3997.1	-	3997.1
	B4	-	4775.0	**-56857.68**	9387.4
	B5	-	6268.2	-	6268.0
Set 4	**C0**	**-82128.14**	3273.6	-	3387.0
	C1	-	4290.5	-	4290.1
	C2	-	5251.9	-	5251.1
	C3	-	5738.1	-	5737.0

Table 4.8. Q-formulation and STP-formulation comparison for IPOPT.

		IPOPT Q		**IPOPT STP**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**
Set 1	**AST3**	**-559.62**	0.5	-50.74	4.2
	R19	**-4524.18**	1.8	-3222.13	131.1
	R19AST3_1	-96.54	387.1	**-3347.01**	387.1
	R18R17_1	**-1981.82**	2.3	-706.05	765.1
	R18R16_1	-2858.50	5.6	-2858.50	11.2
	R17R16_1	**-3462.00**	2.4	-	248.1
	R19AST3_2	-5083.80	97.8	-5083.80	449.1
	R18R17_2	**-2066.86**	107.3	-735.95	142.1
	R18R16_2	**-2920.13**	1710.1	-2536.20	1805.3
	R17R16_2	**-3462.00**	33.7	-3313.86	78.2
	R19AST3_3	**-4169.48**	25.0	-2812.47	489.1
	R18R17_3	**-1121.52**	46.0	-920.84	737.2
	R18R16_3	**-1830.31**	2076.3	-1365.47	2076.2
	R17R16_3	-2500.95	32.4	**-2625.88**	69.6
Set 2	**A0**	-	94.8	**-34166.73**	851.1
	A1	-	73.1	**-27120.26**	789.1
	A2	-	856.0	**-16246.64**	855.3
	A3	-	145.7	**-38012.27**	1061.2
	A4	-	1094.5	**-39574.85**	1093.2
	A5	-	721.0	-	1261.1
	A6	**-42042.98**	55.3	-	1355.2
	A7	**-29364.48**	1423.8	-	1486.3
	A8	**-30333.09**	261.9	-28844.89	1309.3
	A9	**-21304.25**	433.2	-	1526.6
Set 3	**B0**	-	1283.6	**-3420.20**	2139.7
	B1	-	2672.0	-	2639.6
	B2	-	3682.3	-	3664.4
	B3	-	4010.8	-	4000.2
	B4	**-59380.30**	2848.4	-	4781.1
	B5	-	6292.3	-	6313.9
Set 4	**C0**	-	3459.8	-	3632.9
	C1	-	4930.3	**-28660.83**	14153.4
	C2	-	5625.5	**-14187.71**	8036.5
	C3	-	5894.0	-	5779.8

Table 4.9. Q-formulation and STP-formulation comparison for BARON.

		BARON Q			**BARON STP**		
Set	**Problem**	**LB**	**UB**	**Time**	**LB**	**UB**	**Time**
Set 1	**AST3**	-857.11	-561.05	51.1	-	-561.05	0.5
	R19	-5622.02	-4524.18	131.1	-	-4524.18	3.6
	R19AST3_1	-6852.21	-5085.23	387.1	-	-5085.23	80.3
	R18R17_1	-3366.70	-1981.82	765.1	-	-1981.82	4.2
	R18R16_1	-4416.90	-2858.50	1592.1	-	-2858.50	13.1
	R17R16_1	-5099.48	-3462.00	248.1	-	-3462.00	12.9
	R19AST3_2	-6853.95	-5094.18	449.1	-	**-5103.97**	20.6
	R18R17_2	-3665.97	-2188.92	781.1	-	-2188.92	11.1
	R18R16_2	-4451.54	-2971.68	1805.1	-	-2971.68	25.7
	R17R16_2	-5397.76	-3523.62	296.1	-	-3523.62	13.9
	R19AST3_3	-5114.83	-4137.05	489.1	-	**-4169.48**	43.8
	R18R17_3	-2217.47	-1319.13	737.1	-	-1319.13	20.6
	R18R16_3	-2753.86	-2346.80	2076.2	-	-2346.80	36.1
	R17R16_3	-3980.91	-2913.58	348.1	-2978.60	-2913.58	348.1
Set 2	**A0**	-90728.40	**-27226.20**	851.1	-37343.70	-20510.68	851.2
	A1	-67066.54	-14525.16	789.1	-30355.30	**-24890.62**	789.2
	A2	-61934.95	-12574.81	855.2	-23330.00	**-17917.33**	855.2
	A3	-87856.75	**-29922.67**	1061.2	-40761.00	-18171.21	1061.2
	A4	-78196.57	**-23115.18**	1093.2	-42928.50	-17021.58	1093.2
	A5	-67494.19	**-15977.12**	1191.3	-28257.80	-6224.06	1191.3
	A6	-69358.50	**-15647.51**	1355.3	-42463.00	-5405.46	1355.3
	A7	-84539.77	**-26499.75**	1486.4	-44682.20	-10221.20	1486.4
	A8	-70876.13	-18311.77	1309.4	-30666.90	**-19476.27**	1309.4
	A9	-65281.72	**-18203.73**	1526.4	-21934.00	-13436.57	1562.5
Set 3	**B0**	-102125.43	**-9814.85**	2135.4	-45377.30	-8883.27	2135.5
	B1	-127259.71	**-26256.96**	2635.6	-65241.80	-7933.14	2639.1
	B2	-103699.78	**-8898.13**	3626.7	-56320.20	-5155.73	3625.0
	B3	-113638.00	-	3998.2	-74050.50	**-15858.95**	3998.3
	B4	-114518.49	**-318.06**	4785.9	-59469.70	0.00	4776.8
	B5	-122845.00	-	6270.1	-60696.40	-	6270.1
Set 4	**C0**	-195498.75	**-18965.68**	3388.0	-98253.60	-7744.63	3388.3
	C1	-237127.76	-5967.00	4291.6	-119006.00	**-15693.06**	4292.1
	C2	-244485.00	-	5253.2	-136398.61	**-3227.02**	5253.4
	C3	-245821.00	-	5739.7	-130315.02	-	5863.1

objective values. Among the instances KNITRO was better, four belongs to set 2 and one to set 4. Also, it is important to note that for 10 out of 20 instances in set 2, set 3 and set 4, KNITRO failed to find a feasible solution. However, it can be said that, for the cases where it finds a feasible solution, run times of KNITRO are considerably lower than the proposed PSO-based method's, especially for set 1. A similar conclusion can be derived for solution with KNITRO for STP-formulation. In this case, only the result of case B4 is better than the PSO-based method's results. KNITRO terminated without a feasible solution for 12 of the 20 set 2, set 3 and set 4; and run times where feasible solution can be found are notably shorter for set 1 problems.

Comparison between PSO-based method and IPOPT is not much different than the previous comparison in terms of the general picture. For 28 instances, solutions of Q-formulation with IPOPT were inferior and for 29 cases STP-formulation with IPOPT was outperformed by PSO-based method. IPOPT was unable to find a feasible solution for 15 of the 20 instances in set 2, set 3 and set 4 when Q-formulation is used. When STP-formulation is used, one case terminated without a feasible solution for set 1 and 11 cases from the others. Run times of IPOPT are usually shorter than PSO-based method where a feasible solution found for Q-formulation. However, the run is terminated due to time limit in most of the cases where STP-formulation is used.

In almost all of the cases in set 2, set 3 and set 4, BARON was unable to beat the PSO-based method; only the A2 instance in SPT-formulation was superior. BARON yielded better results for four problems when Q-formulation is used, and six problems when STP-formulation is used. BARON was unable to find a feasible solution for four cases with Q-formulation and for two cases with STP-formulation, which are in set 3 and set 4. BARON was able to find optimal solutions for 13 of the 14 instances in set 1 and terminated in a significantly shorter time compared to PSO-based procedure. Among the instances whose optimality is proven, PSO-based method was able find eight of them. KNITRO found three optimal results in Q-formulation and two in STP-formulation. IPOPT was able to find five optimal values in Q-formulation but only one in STP-formulation. Lastly, BARON with Q-formulation found 11 optimal results.

Table 4.10. General purpose solvers and PSO-based method comparison for set 1.

		KNITRO Q		**KNITRO STP**		**IPOPT Q**		**IPOPT STP**		**BARON Q**			**BARON STP**			**PSO-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**LB**	**UB**	**Time**	**LB**	**UB**	**Time**	**Objective**	**Time**
	AST3	-559.62	0.1	-65.00	0.1	-559.62	0.5	-50.74	4.2	-857.11	-561.05	51.1	-	-561.05	0.5	-561.05	51.0
	R19	-4524.18	0.2	-4524.18	94.5	-4524.18	1.8	-3222.13	131.1	-5622.02	-4524.18	131.1	-	-4524.18	3.6	-4524.18	131.3
	R19AST3_1	-5083.80	47.6	-5083.80	2.8	-96.54	387.1	-3347.01	387.1	-6852.21	-5085.23	387.1	-	-5085.23	80.3	-5085.23	387.4
	R18R17_1	-1981.82	0.6	-1429.16	0.2	-1981.82	2.3	-706.05	765.1	-3366.70	-1981.82	765.1	-	-1981.82	4.2	-1981.82	765.1
	R18R16_1	-2841.67	0.4	-2858.50	0.4	-2858.50	5.6	-2858.50	11.2	-4416.90	-2858.50	1592.1	-	-2858.50	13.1	-2858.50	1592.4
	R17R16_1	-3462.00	0.3	-3322.00	0.3	-3462.00	2.4	-	248.1	-5099.48	-3462.00	248.1	-	-3462.00	12.9	-3462.00	248.3
Set 1	**R19AST3_2**	-4594.18	1.5	-4594.18	1.0	**-5083.80**	97.8	**-5083.80**	449.1	-6853.95	**-5094.18**	449.1	-	**-5103.97**	20.6	-5061.06	449.2
	R18R17_2	-2034.16	0.6	-1429.16	0.7	-2066.86	107.3	-735.95	142.1	-3665.97	-2188.92	781.1	-	-2188.92	11.1	-2188.92	781.5
	R18R16_2	-2841.67	0.8	-2841.67	8.1	-2920.13	1710.1	-2536.20	1805.3	-4451.54	-2971.68	1805.1	-	-2971.68	25.7	-2971.68	1805.9
	R17R16_2	-3506.79	1.6	-3322.00	0.3	-3462.00	33.7	-3313.86	78.2	-5397.76	**-3523.62**	296.1	-	**-3523.62**	13.9	-3506.79	296.8
	R19AST3_3	-3779.50	3.6	-4078.44	1.8	**-4169.48**	25.0	-2812.47	489.1	-5114.83	-4137.05	489.1	-	**-4169.48**	43.8	-4167.50	489.3
	R18R17_3	-830.32	1.2	-91.51	0.1	**-1121.52**	46.0	-920.84	737.2	-2217.47	**-1319.13**	737.1	-	**-1319.13**	20.6	-980.36	737.8
	R18R16_3	-1715.28	3.3	-865.75	0.2	-1830.31	2076.3	-1365.47	2076.2	-2753.86	**-2346.80**	2076.2	-	**-2346.80**	36.1	-2343.41	2076.5
	R17R16_3	-2602.82	14.2	-2624.86	1.3	-2500.95	32.4	-2625.88	69.6	-3980.91	-2913.58	348.1	-2978.60	-2913.58	348.1	-2913.58	348.9

Table 4.11. General purpose solvers and PSO-based method comparison for set 2, set 3 and set 4.

		KNITRO Q		KNITRO STP		IPOPT Q		IPOPT STP		BARON Q			BARON STP			PSO-based	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**LB**	**UB**	**Time**	**LB**	**UB**	**Time**	**Objective**	**Time**
Set 2	**A0**	**-33838.65**	241.5	-27939.13	522.9	-	94.8	**-34166.73**	851.1	-90728.40	-27226.20	851.1	-37343.70	-20510.68	851.2	-33469.65	851.1
	A1	-23577.65	8.9	-20668.93	58.8	-	73.1	**-27120.26**	789.1	-67066.54	-14525.16	789.1	-30355.30	-24890.62	789.2	-26508.80	789.5
	A2	-	855.0	-6443.82	50.4	-	856.0	-16246.64	855.3	-61934.95	-12574.81	855.2	-23330.00	**-17917.33**	855.2	-16882.24	855.9
	A3	-33493.38	212.8	-34844.11	102.7	-	145.7	**-38012.27**	1061.2	-87856.75	-29922.67	1061.2	-40761.00	-18171.21	1061.2	-37889.26	1061.7
	A4	**-39656.81**	400.2	-	1093.2	-	1094.5	**-39574.85**	1093.2	-78196.57	-23115.18	1093.2	-42928.50	-17021.58	1093.2	-38320.45	1093.7
	A5	**-26265.05**	1252.9	-	1191.0	-	721.0	-	1261.1	-67494.19	-15977.12	1191.3	-28257.80	-6224.06	1191.3	-25372.95	1191.3
	A6	**-41945.96**	515.9	-41906.40	1104.9	**-42042.98**	55.3	-	1355.2	-69358.50	-15647.51	1355.3	-42463.00	-5405.46	1355.3	-41941.17	1355.1
	A7	-	1486.0	-	1486.0	-29364.48	1423.8	-	1486.3	-84539.77	-26499.75	1486.4	-44682.20	-10221.20	1486.4	-42838.33	1487.0
	A8	-	1309.0	-	1309.0	**-30333.09**	261.9	-28844.89	1309.3	-70876.13	-18311.77	1309.4	-30666.90	-19476.27	1309.4	-30249.15	1309.8
	A9	-21593.58	943.2	-20774.54	1249.5	-21304.25	433.2	-	1526.6	-65281.72	-18203.73	1526.4	-21934.00	-13436.57	1562.5	-21689.53	1526.5
Set 3	**B0**	-37210.20	91.6	-	2135.0	-	1283.6	-3420.20	2139.7	-102125.43	-9814.85	2135.4	-45377.30	-8883.27	2135.5	-40150.09	2135.9
	B1	-57932.09	1292.8	**-58299.34**	2465.6	-	2672.0	-	2639.6	-127259.71	-26256.96	2635.6	-65241.80	-7933.14	2639.1	-57999.63	2635.1
	B2	-	3624.0	-	3624.6	-	3682.3	-	3664.4	-103699.78	-8898.13	3626.7	-56320.20	-5155.73	3625.0	-49892.58	3624.5
	B3	-	3997.1	-	3997.1	-	4010.8	-	4000.2	-113638.00	-	3998.2	-74050.50	-15858.95	3998.3	-72861.33	3997.7
	B4	-	4775.0	-56857.68	9387.4	**-59380.30**	2848.4	-	4781.1	-114518.49	-318.06	4785.9	-59469.70	0.00	4776.8	-58759.30	4775.8
	B5	-	6268.2	-	6268.0	-	6292.3	-	6313.9	-122845.00	-	6270.1	-60696.40	-	6270.1	-60060.83	6268.5
Set 4	**C0**	**-82128.14**	3273.6	-	3387.0	-	3459.8	-	3632.9	-195498.75	-18965.68	3388.0	-98253.60	-7744.63	3388.3	-66337.18	3387.5
	C1	-	4290.5	-	4290.1	-	4930.3	-28660.83	14153.4	-237127.76	-5967.00	4291.6	-119006.00	-15693.06	4292.1	-81091.98	4290.9
	C2	-	5251.9	-	5251.1	-	5625.5	-14187.71	8036.5	-244485.00	-	5253.2	-136398.61	-3227.02	5253.4	-111315.98	5251.5
	C3	-	5738.1	-	5737.0	-	5894.0	-	5779.8	-245821.00	-	5739.7	-130315.02	-	5863.1	-109249.43	5737.5

Table 4.12 displays the pair-wise comparison between PSO-based method and other heuristics, namely GCH, MALT and VNS. If an objective value is strictly better than the PSO-based method then it is shown in bold. When Table 4.12 is examined, it can be said that GCH was unable to find any better solution and the comparison against MALT procedure shows that for only two of the cases MALT was able to find a better solution than PSO-based procedure. When VNS is compared, it is seen that the PSO-based solution method dominates all of the instances in set 1. For the other sets, PSO-based method was superior in 12 of the cases out of 20; seven in set 2, three in set 3 and two in set 4.

The comparison between the performances of SA-based method and the general purpose solvers are shown in the Table 4.13 for set 1 and Table 4.14 for the remaining. Similar to previous comparisons, pair-wise comparisons are conducted between each of the solver solutions and the SA-based solutions. Likewise, if an objective value is strictly better than the SA-based method, then it is presented in bold.

The comparison between the Q-formulation solved with KNITRO and SA-based methods shows that in five of all of the instances KNITRO performed better than proposed SA-based method. Like before, run times of KNITRO is considerably lower than the proposed SA-based method's for the cases where it finds a feasible solution and especially for set 1. Similar results can be seen in comparison to KNITRO for STP-formulation. In this case, only two of the results out of 34 are better than the SA-based method's results. Again, run times are usually shorter than SA-based method for the cases where a feasible solution is found.

In parallel to the previous comparisons, solutions of Q-formulation with IPOPT were inferior for 24 cases. IPOPT managed to be superior to SA-based approach only in three cases for STP-formulation. Like before, run times of IPOPT are usually shorter than proposed SA-based method where a feasible solution found for Q-formulation, but, it is not the case when STP-formulation is used.

Table 4.12. Heuristic methods and PSO-based method comparison.

		GCH		**MALT**		**VNS**		**PSO-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**
Set 1	**AST3**	-552.85	4.6	-557.49	51.0	-554.16	51.1	-561.05	51.0
	R19	-4513.39	110.1	-4524.18	131.0	-4524.18	131.4	-4524.18	131.3
	R19AST3_1	-4480.17	212.4	-5079.32	387.1	-5085.23	387.6	-5085.23	387.4
	R18R17_1	-1280.10	481.6	-1981.82	765.1	-1512.16	765.5	-1981.82	765.1
	R18R16_1	-2767.00	1475.0	-2858.50	1592.5	-2858.50	1593.6	-2858.50	1592.4
	R17R16_1	-3462.00	248.8	-3452.98	248.1	-2992.34	248.4	-3462.00	248.3
	R19AST3_2	-4258.38	420.7	**-5067.37**	449.1	-4524.18	449.7	-5061.06	449.2
	R18R17_2	-1338.82	298.9	-2047.76	781.0	-1443.92	782.1	-2188.92	781.5
	R18R16_2	-2813.09	709.8	-2951.19	1805.2	-2901.31	1807.1	-2971.68	1805.9
	R17R16_2	-3343.13	121.6	-3445.54	296.0	-3445.17	297.0	-3506.79	296.8
	R19AST3_3	-2936.64	80.8	-4083.88	489.1	-3735.23	489.7	-4167.50	489.3
	R18R17_3	-680.10	284.6	-689.16	737.1	-689.16	738.2	-980.36	737.8
	R18R16_3	-1829.63	1918.5	-2325.46	2076.2	-2302.20	2078.5	-2343.41	2076.5
	R17R16_3	-2273.80	167.6	-2762.73	348.1	-2861.56	349.1	-2913.58	348.9
Set 2	**A0**	-18464.33	184.1	-32820.76	851.1	-33067.42	863.1	-33469.65	851.1
	A1	-6282.29	119.2	-16583.12	789.0	-17553.96	790.8	-26508.80	789.5
	A2	-6699.04	127.6	**-19422.44**	855.0	-14104.50	857.1	-16882.24	855.9
	A3	-24417.47	711.9	-35407.84	1061.1	**-37890.59**	1066.7	-37889.26	1061.7
	A4	-27163.24	323.6	-37532.21	1093.5	-37367.87	1100.0	-38320.45	1093.7
	A5	-19602.47	489.2	-23746.37	1191.2	-24912.02	1196.2	-25372.95	1191.3
	A6	-22484.50	419.6	-41330.57	1355.3	**-41981.83**	1360.4	-41941.17	1355.1
	A7	-23635.36	324.1	-41370.72	1486.7	-42360.35	1490.1	-42838.33	1487.0
	A8	-17951.92	364.2	-24242.31	1309.1	-29415.77	1319.2	-30249.15	1309.8
	A9	-16050.83	412.7	-20299.78	1526.5	**-21784.73**	1532.0	-21689.53	1526.5
Set 3	**B0**	-19601.09	553.2	-34409.74	2135.3	-39926.02	2142.1	-40150.09	2135.9
	B1	-27876.72	552.7	-54850.38	2635.4	-57977.08	2644.0	-57999.63	2635.1
	B2	-28434.78	1266.6	-48887.33	3624.4	**-50896.49**	3637.5	-49892.58	3624.5
	B3	-43721.24	837.2	-67560.29	3997.1	-68661.65	4002.2	-72861.33	3997.7
	B4	-32347.64	1224.1	-53647.78	4775.5	**-59208.37**	4777.6	-58759.30	4775.8
	B5	-32686.58	1340.9	-50159.38	6268.3	**-60334.21**	6276.4	-60060.83	6268.5
Set 4	**C0**	-22796.34	722.8	-64243.97	3390.0	**-68265.23**	3404.9	-66337.18	3387.5
	C1	-34054.62	629.3	-68427.27	4307.3	**-81572.92**	4355.1	-81091.98	4290.9
	C2	-50703.12	1394.4	-96241.47	5265.4	-109320.80	5278.6	-111315.98	5251.5
	C3	-30767.65	1268.3	-93697.17	5740.4	-104596.01	5793.5	-109249.43	5737.5

Table 4.13. General purpose solvers and SA-based method comparison for set 1.

		KNITRO Q		**KNITRO STP**		**IPOPT Q**		**IPOPT STP**		**BARON Q**			**BARON STP**			**SA-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**LB**	**UB**	**Time**	**LB**	**UB**	**Time**	**Objective**	**Time**
	AST3	-559.62	0.1	-65.00	0.1	-559.62	0.5	-50.74	4.2	-857.11	**-561.05**	51.1	-	**-561.05**	0.5	-560.95	17.4
	R19	-4524.18	0.2	-4524.18	94.5	-4524.18	1.8	-3222.13	131.1	-5622.02	-4524.18	131.1	-	-4524.18	3.6	-4524.18	102.4
	R19AST3_1	-5083.80	47.6	-5083.80	2.8	-96.54	387.1	-3347.01	387.1	-6852.21	**-5085.23**	387.1	-	**-5085.23**	80.3	-5083.80	316.0
	R18R17_1	-1981.82	0.6	-1429.16	0.2	-1981.82	2.3	-706.05	765.1	-3366.70	-1981.82	765.1	-	-1981.82	4.2	-1981.82	481.7
	R18R16_1	-2841.67	0.4	-2858.50	0.4	-2858.50	5.6	-2858.50	11.2	-4416.90	-2858.50	1592.1	-	-2858.50	13.1	-2858.50	1213.5
	R17R16_1	-3462.00	0.3	-3322.00	0.3	-3462.00	2.4	-	248.1	-5099.48	-3462.00	248.1	-	-3462.00	12.9	-3462.00	175.3
Set 1	**R19AST3_2**	-4594.18	1.5	-4594.18	1.0	**-5083.80**	97.8	**-5083.80**	449.1	-6853.95	**-5094.18**	449.1	-	**-5103.97**	20.6	-4594.18	346.8
	R18R17_2	-2034.16	0.6	-1429.16	0.7	-2066.86	107.3	-735.95	142.1	-3665.97	-2188.92	781.1	-	-2188.92	11.1	-2188.92	625.6
	R18R16_2	-2841.67	0.8	-2841.67	8.1	-2920.06	1710.1	-2536.20	1805.3	-4451.54	**-2971.68**	1805.1	-	**-2971.68**	25.7	-2920.06	1348.5
	R17R16_2	**-3506.79**	1.6	-3322.00	0.3	**-3462.00**	33.7	-3313.86	78.2	-5397.76	**-3523.62**	296.1	-	**-3523.62**	13.9	-3460.53	202.7
	R19AST3_3	-3779.50	3.6	**-4078.44**	1.8	**-4169.48**	25.0	-2812.47	489.1	-5114.83	**-4137.05**	489.1	-	**-4169.48**	43.8	-4077.03	403.7
	R18R17_3	-830.32	1.2	-91.51	0.1	**-1121.52**	46.0	-920.84	737.2	-2217.47	**-1319.13**	737.1	-	**-1319.13**	20.6	-980.36	555.0
	R18R16_3	-1715.28	3.3	-865.75	0.2	-1830.31	2076.3	-1365.47	2076.2	-2753.86	**-2346.80**	2076.2	-	**-2346.80**	36.1	-2333.99	1523.4
	R17R16_3	-2602.82	14.2	-2624.86	1.3	-2500.95	32.4	-2625.88	69.6	-3980.91	**-2913.58**	348.1	-2978.60	**-2913.58**	348.1	-2910.92	308.4

Table 4.14. General purpose solvers and SA-based method comparison for set 2, set 3 and set 4.

		KNITRO Q		**KNITRO STP**		**IPOPT Q**		**IPOPT STP**		**BARON Q**			**BARON STP**			**SA-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**LB**	**UB**	**Time**	**LB**	**UB**	**Time**	**Objective**	**Time**
Set 2	**A0**	-33838.65	241.5	-27939.13	522.9	-	94.8	-34166.73	851.1	-90728.40	-27226.20	851.1	-37343.70	-20510.68	851.2	-34564.00	859.4
	A1	**-23577.65**	8.9	**-20668.93**	58.8	-	73.1	**-27120.26**	789.1	-67066.54	-14525.16	789.1	-30355.30	**-24890.62**	789.2	-19324.58	589.3
	A2	-	855.0	-6443.82	50.4	-	856.0	-16246.64	855.3	-61934.95	-12574.81	855.2	-23330.00	-17917.33	855.2	-18389.39	662.0
	A3	-33493.38	212.8	-34844.11	102.7	-	145.7	-38012.27	1061.2	-87856.75	-29922.67	1061.2	-40761.00	-18171.21	1061.2	-38237.24	1064.2
	A4	**-39656.81**	400.2	-	1093.2	-	1094.5	**-39574.85**	1093.2	-78196.57	-23115.18	1093.2	-42928.50	-17021.58	1093.2	-38968.48	1099.4
	A5	**-26265.05**	1252.9	-	1191.0	-	721.0	-	1261.1	-67494.19	-15977.12	1191.3	-28257.80	-6224.06	1191.3	-25331.73	1201.7
	A6	-41945.96	515.9	-41906.40	1104.9	**-42042.98**	55.3	-	1355.2	-69358.50	-15647.51	1355.3	-42463.00	-5405.46	1355.3	-41995.98	1363.4
	A7	-	1486.0	-	1486.0	-29364.48	1423.8	-	1486.3	-84539.77	-26499.75	1486.4	-44682.20	-10221.20	1486.4	-43260.04	1493.3
	A8	-	1309.0	-	1309.0	**-30333.09**	261.9	-28844.89	1309.3	-70876.13	-18311.77	1309.4	-30666.90	-19476.27	1309.4	-29057.56	1315.1
	A9	-21593.58	943.2	-20774.54	1249.5	-21304.25	433.2	-	1526.6	-65281.72	-18203.73	1526.4	-21934.00	-13436.57	1562.5	-21851.79	1541.9
Set 3	**B0**	-37210.20	91.6	-	2135.0	-	1283.6	-3420.20	2139.7	-102125.43	-9814.85	2135.4	-45377.30	-8883.27	2135.5	-41425.54	1457.3
	B1	-57932.09	1292.8	-58299.34	2465.6	-	2672.0	-	2639.6	-127259.71	-26256.96	2635.6	-65241.80	-7933.14	2639.1	-59163.90	1929.4
	B2	-	3624.0	-	3624.6	-	3682.3	-	3664.4	-103699.78	-8898.13	3626.7	-56320.20	-5155.73	3625.0	-51749.24	3012.6
	B3	-	3997.1	-	3997.1	-	4010.8	-	4000.2	-113638.00	-	3998.2	-74050.50	-15858.95	3998.3	-72886.45	3712.3
	B4	-	4775.0	-56857.68	9387.4	**-59380.30**	2848.4	-	4781.1	-114518.49	-318.06	4785.9	-59469.70	0.00	4776.8	-59293.62	4803.0
	B5	-	6268.2	-	6268.0	-	6292.3	-	6313.9	-122845.00	-	6270.1	-60696.40	-	6270.1	-60150.66	5605.0
Set 4	**C0**	**-82128.14**	3273.6	-	3387.0	-	3459.8	-	3632.9	-195498.75	-18965.68	3388.0	-98253.60	-7744.63	3388.3	-75614.89	2831.9
	C1	-	4290.5	-	4290.1	-	4930.3	-28660.83	14153.4	-237127.76	-5967.00	4291.6	-119006.00	-15693.06	4292.1	-87982.64	3636.8
	C2	-	5251.9	-	5251.1	-	5625.5	-14187.71	8036.5	-244485.00	-	5253.2	-136398.61	-3227.02	5253.4	-117096.68	4971.2
	C3	-	5738.1	-	5737.0	-	5894.0	-	5779.8	-245821.00	-	5739.7	-130315.02	-	5863.1	-111244.69	5742.5

As in the comparison to PSO-based method, SA-based method dominated BARON in set 2, set 3 and set 4; except for the A2 instance in SPT-formulation. However, BARON yielded better results for nine problems for both formulations in set 1. The run times for BARON are again significantly smaller for the cases where an optimal solution is found. SA-based method was able to solve five of the problems that have proven optimal solutions to optimality.

Table 4.15 displays the pair-wise comparison between SA-based method and GCH, MALT and VNS. Like before, strictly better objective values than the SA-based method are shown in bold. Table 4.15 states that GCH was unable to find any better solution again and MALT was able to find a better solution than SA-based procedure for four instances among all 34, where three of them are in set 1 and the other one in set 2. SA-based procedure dominated VNS in all instances except one instance each in set 1, set 2 and set 3.

Table 4.16 provides a one to one comparison between the two proposed solution methods. For set 1, PSO-based method was superior in eight of the instances and both methods performed the same for the rest. For set 2, in three problems PSO-based method, in seven problems SA-based method performed better. Lastly, for all of the instances of set 3 and set 4 SA-based approach was the dominating one. Also, SA-based method terminated in 24 out of 34 instances before time limit is reached. In total, 56512.2 seconds spent for SA-based method calculations whereas 63787.9 seconds spent for PSO-based method.

In order to be able to say more about the solution quality of the proposed methods, gaps are calculated using the Equation 4.1. Lower bounds of BARON with STP-formulation (LB_{STP}) are used and gaps between the two proposed methods as well as the best gaps obtained among all solution techniques are presented in Table 4.17. In general, it can be said that reasonable gaps are achieved for set 1, set 2 and set 3 problems whereas it is not exactly the case for set 4. However, this can be due to poor

Table 4.15. Heuristic methods and SA-based method comparison.

		GCH		**MALT**		**VNS**		**SA-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**	**Objective**	**Time**
Set 1	**AST3**	-552.85	4.6	-557.49	51.0	-554.16	51.1	-560.95	17.4
	R19	-4513.39	110.1	-4524.18	131.0	-4524.18	131.4	-4524.18	102.4
	R19AST3_1	-4480.17	212.4	-5079.32	387.1	**-5085.23**	387.6	-5083.80	316.0
	R18R17_1	-1280.10	481.6	-1981.82	765.1	-1512.16	765.5	-1981.82	481.7
	R18R16_1	-2767.00	1475.0	-2858.50	1592.5	-2858.50	1593.6	-2858.50	1213.5
	R17R16_1	-3462.00	248.8	-3452.98	248.1	-2992.34	248.4	-3462.00	175.3
	R19AST3_2	-4258.38	420.7	**-5067.37**	449.1	-4524.18	449.7	-4594.18	346.8
	R18R17_2	-1338.82	298.9	-2047.76	781.0	-1443.92	782.1	-2188.92	625.6
	R18R16_2	-2813.09	709.8	**-2951.19**	1805.2	-2901.31	1807.1	-2920.06	1348.5
	R17R16_2	-3343.13	121.6	-3445.54	296.0	-3445.17	297.0	-3460.53	202.7
	R19AST3_3	-2936.64	80.8	**-4083.88**	489.1	-3735.23	489.7	-4077.03	403.7
	R18R17_3	-680.10	284.6	-689.16	737.1	-689.16	738.2	-980.36	555.0
	R18R16_3	-1829.63	1918.5	-2325.46	2076.2	-2302.20	2078.5	-2333.99	1523.4
	R17R16_3	-2273.80	167.6	-2762.73	348.1	-2861.56	349.1	-2910.92	308.4
Set 2	**A0**	-18464.33	184.1	-32820.76	851.1	-33067.42	863.1	-34564.00	859.4
	A1	-6282.29	119.2	-16583.12	789.0	-17553.96	790.8	-19324.58	589.3
	A2	-6699.04	127.6	**-19422.44**	855.0	-14104.50	857.1	-18389.39	662.0
	A3	-24417.47	711.9	-35407.84	1061.1	-37890.59	1066.7	-38237.24	1064.2
	A4	-27163.24	323.6	-37532.21	1093.5	-37367.87	1100.0	-38968.48	1099.4
	A5	-19602.47	489.2	-23746.37	1191.2	-24912.02	1196.2	-25331.73	1201.7
	A6	-22484.50	419.6	-41330.57	1355.3	-41981.83	1360.4	-41995.98	1363.4
	A7	-23635.36	324.1	-41370.72	1486.7	-42360.35	1490.1	-43260.04	1493.3
	A8	-17951.92	364.2	-24242.31	1309.1	**-29415.77**	1319.2	-29057.56	1315.1
	A9	-16050.83	412.7	-20299.78	1526.5	-21784.73	1532.0	-21851.79	1541.9
Set 3	**B0**	-19601.09	553.2	-34409.74	2135.3	-39926.02	2142.1	-41425.54	1457.3
	B1	-27876.72	552.7	-54850.38	2635.4	-57977.08	2644.0	-59163.90	1929.4
	B2	-28434.78	1266.6	-48887.33	3624.4	-50896.49	3637.5	-51749.24	3012.6
	B3	-43721.24	837.2	-67560.29	3997.1	-68661.65	4002.2	-72886.45	3712.3
	B4	-32347.64	1224.1	-53647.78	4775.5	-59208.37	4777.6	-59293.62	4803.0
	B5	-32686.58	1340.9	-50159.38	6268.3	**-60334.21**	6276.4	-60150.66	5605.0
Set 4	**C0**	-22796.34	722.8	-64243.97	3390.0	-68265.23	3404.9	-75614.89	2831.9
	C1	-34054.62	629.3	-68427.27	4307.3	-81572.92	4355.1	-87982.64	3636.8
	C2	-50703.12	1394.4	-96241.47	5265.4	-109320.80	5278.6	-117096.68	4971.2
	C3	-30767.65	1268.3	-93697.17	5740.4	-104596.01	5793.5	-111244.69	5742.5

Table 4.16. PSO-based and SA-based method comparison.

		PSO-based		**SA-based**	
Set	**Problem**	**Objective**	**Time**	**Objective**	**Time**
Set 1	**AST3**	**-561.05**	51.0	-560.95	17.4
	R19	-4524.18	131.3	-4524.18	102.4
	R19AST3_1	**-5085.23**	387.4	-5083.80	316.0
	R18R17_1	-1981.82	765.1	-1981.82	481.7
	R18R16_1	-2858.50	1592.4	-2858.50	1213.5
	R17R16_1	-3462.00	248.3	-3462.00	175.3
	R19AST3_2	**-5061.06**	449.2	-4594.18	346.8
	R18R17_2	-2188.92	781.5	-2188.92	625.6
	R18R16_2	**-2971.68**	1805.9	-2920.06	1348.5
	R17R16_2	**-3506.79**	296.8	-3460.53	202.7
	R19AST3_3	**-4167.50**	489.3	-4077.03	403.7
	R18R17_3	-980.36	737.8	-980.36	555.0
	R18R16_3	**-2343.41**	2076.5	-2333.99	1523.4
	R17R16_3	**-2913.58**	348.9	-2910.92	308.4
Set 2	**A0**	-33469.65	851.1	**-34564.00**	859.4
	A1	**-26508.80**	789.5	-19324.58	589.3
	A2	-16882.24	855.9	**-18389.39**	662.0
	A3	-37889.26	1061.7	**-38237.24**	1064.2
	A4	-38320.45	1093.7	**-38968.48**	1099.4
	A5	**-25372.95**	1191.3	-25331.73	1201.7
	A6	-41941.17	1355.1	**-41995.98**	1363.4
	A7	-42838.33	1487.0	**-43260.04**	1493.3
	A8	**-30249.15**	1309.8	-29057.56	1315.1
	A9	-21689.53	1526.5	**-21851.79**	1541.9
Set 3	**B0**	-40150.09	2135.9	**-41425.54**	1457.3
	B1	-57999.63	2635.1	**-59163.90**	1929.4
	B2	-49892.58	3624.5	**-51749.24**	3012.6
	B3	-72861.33	3997.7	**-72886.45**	3712.3
	B4	-58759.30	4775.8	**-59293.62**	4803.0
	B5	-60060.83	6268.5	**-60150.66**	5605.0
Set 4	**C0**	-66337.18	3387.5	**-75614.89**	2831.9
	C1	-81091.98	4290.9	**-87982.64**	3636.8
	C2	-111315.98	5251.5	**-117096.68**	4971.2
	C3	-109249.43	5737.5	**-111244.69**	5742.5

lower bounds for the instances in set 4 as well as poor solution quality.

$$gap = \left| \frac{objective - LB_{STP}}{objective} \right| 100 \tag{4.1}$$

When the individual performances of the proposed solution methods are inspected, it can be said that both procedures are dominated by BARON with STP-formulation for the instances in set 1. For set 2, set 3 and set 4 BARON performed poorly, moreover KNITRO and IPOPT turned out to be unreliable since they were unable to find a feasible solution in most of the times. For these three sets, SA-based method performed well in most of the cases. However, especially for set 3 and set 4, the performance of PSO-based method was comparable to the performance of VNS procedure.

Friedman test can be used to see if there is a significant effect of the solution technique used on the solution quality. For this purpose, first all of the solution methods and problem instances are used and Table 4.18 presents the results of this test. Since $p < 0.05$, it can be concluded that there is a significant difference among solution techniques. SA-based, PSO-based and VNS methods end up with the smallest ranks after this test; therefore as a further analysis, Friedman test is performed on those three in pair-wise manner. Tables 4.19, 4.20 and 4.21 show the result of these tests.

Table 4.17. Optimality gap percentages.

Set	Problem	PSO-based Gap	SA-based Gap	Best Gap
Set 1	**AST3**	0	0.0162	0
	R19	0	0	0
	R19AST3_1	0	0.0282	0
	R18R17_1	0	0	0
	R18R16_1	0	0	0
	R17R16_1	0	0	0
	R19AST3_2	0.8478	11.0963	0
	R18R17_2	0	0	0
	R18R16_2	0	1.7678	0
	R17R16_2	0.4800	1.8233	0
	R19AST3_3	0.0476	2.2677	0
	R18R17_3	34.5556	34.5556	0
	R18R16_3	0.1445	0.5490	0
	R17R16_3	2.2317	2.3249	2.2316
Set 2	**A0**	11.5748	8.0422	8.0422
	A1	14.5103	57.0813	11.9285
	A2	38.1925	26.8667	20.1188
	A3	7.5793	6.6003	6.6003
	A4	12.0250	10.1621	8.2500
	A5	11.3698	11.5510	7.5871
	A6	1.2442	1.1121	0.9990
	A7	4.3043	3.2875	3.2875
	A8	1.3810	5.5385	1.1005
	A9	1.1271	0.3762	0.3762
Set 3	**B0**	13.0192	9.5394	9.5394
	B1	12.4866	10.2730	10.2730
	B2	12.8829	8.8329	8.8329
	B3	1.6321	1.5971	1.5971
	B4	1.2090	0.2970	0.1506
	B5	1.0582	0.9073	0.6003
Set 4	**C0**	48.1124	29.9395	19.6345
	C1	46.7543	35.2608	35.2608
	C2	22.5328	16.4838	16.4838
	C3	19.2821	17.1427	17.1427

Table 4.18. Friedman test result for all solution techniques for all instances.

Method	Observation Count	Est Median	Sum of Ranks
BARON-Q	34	-10281	197.0
BARON-STP	34	-9473	191.0
GCH	34	-8886	269.0
IPOPT-Q	34	-11368	225.0
IPOPT-STP	34	-11683	273.5
KNITRO-Q	34	-12352	230.0
KNITRO-STP	34	-11885	283.5
MALT	34	-14372	184.0
PSO-based	34	-15500	112.5
SA-based	34	-15548	111.5
VNS	34	-14904	167.0
p = 0,000 (adjusted for ties)			

Table 4.19. Friedman test result for PSO-based and VNS methods for all instances.

Method	Observation Count	Est Median	Sum of Ranks
PSO-based	34	-23661	44.0
VNS	34	-23513	58.0
p = 0.013 (adjusted for ties)			

Table 4.20. Friedman test result for SA-based and VNS methods for all instances.

Method	Observation Count	Est Median	Sum of Ranks
SA-based	34	-23662	38.0
VNS	34	-23278	64.0
p = 0.000 (adjusted for ties)			

Table 4.21. Friedman test result for PSO-based and SA-based methods for all instances.

Method	Observation Count	Est Median	Sum of Ranks
PSO-based	34	-24128	53.0
SA-based	34	-24141	49.0
p = 0.465 (adjusted for ties)			

Tables 4.19, 4.20 and 4.21 indicate that, both PSO-based and SA-based methods perform significantly better than VNS when all problem instances are considered. However there is no significant difference between the two proposed methods, PSO-based and SA-based methods.

Finally, Friedman test is conducted for all solution techniques over the problem sets 2, 3 and 4; since for the instances in set 1 the optimal solutions are found except for one problem. Table 4.22 presents the results of the test results. Since $p < 0.05$, there is again a significant difference among solution techniques and SA-based, PSO-based and VNS methods end up with the smallest ranks. For those three, Friedman test is performed in pair-wise manner and Tables 4.23, 4.24 and 4.25 present the result of these tests.

Table 4.22. Friedman test result for all solution techniques for set 2, set 3 and set 4.

Method	Observation Count	Est Median	Sum of Ranks
BARON-Q	20	-11090	157.5
BARON-STP	20	-7924	155.0
GCH	20	-20952	139.0
IPOPT-Q	20	-3967	163.0
IPOPT-STP	20	-3535	146.5
KNITRO-Q	20	-17045	130.5
KNITRO-STP	20	-6654	162.5
MALT	20	-37417	100.0
PSO-based	20	-39977	60.0
SA-based	20	-40487	37.0
VNS	20	-39458	69.0
p = 0,000 (adjusted for ties)			

Table 4.23. Friedman test result for PSO-based and VNS methods for set 2, set 3 and set 4.

Method	Observation Count	Est Median	Sum of Ranks
PSO-based	20	-42437	28.0
VNS	20	-42124	32.0
p = 0.371			

Table 4.24. Friedman test result for SA-based and VNS methods for set 2, set 3 and set 4.

Method	Observation Count	Est Median	Sum of Ranks
SA-based	20	-43070	22.0
VNS	20	-41729	38.0
p = 0.000			

Table 4.25. Friedman test result for PSO-based and SA-based methods for set 2, set 3 and set 4.

Method	Observation Count	Est Median	Sum of Ranks
PSO-based	20	-42213	37.0
SA-based	20	-42804	23.0
p = 0.002			

Tables 4.23, 4.24 and 4.25 indicate that, SA-based method perform significantly better than both PSO-based method and VNS when sets 2, 3 and 4 are considered. Also, there is no significant difference between PSO-based method and VNS for the large instances.

5. CONCLUSION

In the scope of this thesis, the pooling problem which is a well known bilinear, NP-hard problem is studied. Although there are exact methods to solve the pooling problem in the literature, these techniques are ineffective for large problem instances. There are also heuristic methods that are proposed for the problem, but there is no modern metaheuristic approaches studied in this context except for variable neighborhood search in the literature. In this study, two metaheuristic methods are proposed to solve the pooling problem, namely an approach based on particle swarm optimization and a simulated annealing approach that employs variable neighborhood size. Both methods exploit the bilinear structure of the formulation in new solution generation, fitness calculation and local improvement.

Particle swarm optimization, a swarm based metaheuristic which is originally designed to solve continuous optimization problems, is adapted as the first proposed method. In this method, the problem is mainly reduced to optimization of one of the variable sets and this fixed search variable is subjected to particle swarm operations. Also a fast local optimization heuristic called ALT is incorporated to speed up the convergence of PSO-based procedure. Also, some modifications in parameters are designed to explore the feasible space at the beginning and intensify at the end of the algorithm. As the second metaheuristic approach, simulated annealing which is originally emerged to solve combinatorial problems is modified to solve the pooling problem is studied. The dynamic neighborhood scheme employed in this method is based on a previously used one in a VNS approach for this problem. Also, the same local improvement heuristic is integrated in the SA-based method to improve solution quality.

One of the main advantages of the proposed methods is that they always provide a feasible solution regardless of the iteration. Another advantage is that the SA-based procedure can be run as if the problem has no constraints and PSO-based method only needs to consider the constraints that are entirely composed of the search variable.

To evaluate the performances of these methods, problem instances of different sizes in the literature are used and 12 new ones are fabricated. Open source and commercial general purpose solvers are employed along with the heuristic methods in the literature as benchmark methods. In general, the general purpose global optimizer BARON was able to find the optimum of almost all of the smaller problems that are given as set 1. However, for the larger problem sets, the proposed methods performed better, especially SA-based method dominated the others. Also from an operational perspective, the run times are reasonable for a tactical problem: All solutions are provided under two hours whereas most them lasted under an hour.

Finally, the generalization of the two proposed methods to general bilinear programming problems is a possible research direction as a future work. Both approaches can be generalized for continuous bilinear optimization models in a similar fashion.

APPENDIX A: P-FORMULATION OF THE POOLING PROBLEM

Table A.1. Sets, parameters, and decision variables for P-formulation.

Sets	
I	Set of input streams (raw materials)
L	Set of pools
J	Set of output streams (end products)
K	Set of quality attributes
T_X	Set of (i, l) pairs for which input to pool connection exists
T_Y	Set of (l, j) pairs for which pool to output connection exists
T_Z	Set of (i, j) pairs for which input to output connection exists
Parameters	
c_{il}	Unit cost of raw material i sent to pool l
c_{ij}	Unit cost of raw material i sent to output j
d_j	Unit revenue for product j
A_i^L	Minimum required usage of raw material i
A_i^U	Maximum availability of raw material i
S_l	Capacity of pool l
D_j^L	Minimum required production of end product j
D_j^U	Demand upper limit of end product j
C_{ik}	Level of quality attribute k in raw material i
P_{jk}^L	Lower limit of quality attribute k in end product j
P_{jk}^U	Upper limit of quality attribute k in end product j
Decision variables	
x_{il}	Flow from input stream i to pool l
y_{lj}	Flow from pool l to output j
z_{ij}	Flow from input stream i to output j
p_{lk}	Level of quality attribute k in pool l

P-formulation:

$$\min_{\substack{x_{il},y_{lj},\\ z_{i,j},p_{lk}}} \sum_{(i,l)\in T_X} c_{il}x_{il} - \sum_{(l,j)\in T_Y} d_j y_{lj} - \sum_{(i,j)\in T_Z} (d_j - c_{ij})z_{ij} \tag{A.1}$$

s.t.

$$A_i^L \leq \sum_{l:(i,l)\in T_X} x_{il} + \sum_{j:(i,j)\in T_Z} z_{ij} \leq A_i^U \qquad \forall i \tag{A.2}$$

$$\sum_{i:(i,l)\in T_X} x_{il} \leq S_l \qquad \forall l \tag{A.3}$$

$$D_j^L \leq \sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \leq D_j^U \qquad \forall j \tag{A.4}$$

$$\sum_{i:(i,l)\in T_X} x_{il} - \sum_{j:(l,j)\in T_Y} y_{lj} = 0 \qquad \forall l \tag{A.5}$$

$$\sum_{l:(l,j)\in T_Y} p_{lk}y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}z_{ij} \leq P_{jk}^U \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{A.6}$$

$$\sum_{l:(l,j)\in T_Y} p_{lk}y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}z_{ij} \geq P_{jk}^L \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{A.7}$$

$$\sum_{i:(i,l)\in T_X} C_{ik}x_{il} = p_{lk} \sum_{j:(l,j)\in T_Y} y_{lj} \qquad \forall l,k \tag{A.8}$$

$$0 \leq x_{il} \leq \min \left\{ S_l, A_i^U, \sum_{j:(l,j)\in T_Y} D_j^U \right\} \qquad \forall (i,l)\in T_X \tag{A.9}$$

$$0 \leq y_{lj} \leq \min \left\{ S_l, D_j^U, \sum_{i:(i,l)\in T_X} A_i^U \right\} \qquad \forall (l,j)\in T_Y \tag{A.10}$$

$$0 \leq z_{ij} \leq \min \left\{ A_i^U, D_j^U \right\} \qquad \forall (i,j)\in T_Z \tag{A.11}$$

$$\min_i C_{ik} \leq p_{lk} \leq \max_i C_{ik} \qquad \forall l,k \tag{A.12}$$

APPENDIX B: STP-FORMULATION OF THE POOLING PROBLEM

Table B.1. Sets, parameters, and decision variables for STP-formulation.

Sets	
I	Set of input streams (raw materials)
L	Set of pools
J	Set of output streams (end products)
K	Set of quality attributes
T_X	Set of (i, l) pairs for which input to pool connection exists
T_Y	Set of (l, j) pairs for which pool to output connection exists
T_Z	Set of (i, j) pairs for which input to output connection exists
Parameters	
c_{il}	Unit cost of raw material i sent to pool l
c_{ij}	Unit cost of raw material i sent to output j
d_j	Unit revenue for product j
A_i^L	Minimum required usage of raw material i
A_i^U	Maximum availability of raw material i
S_l	Capacity of pool l
D_j^L	Minimum required production of end product j
D_j^U	Demand upper limit of end product j
C_{ik}	Level of quality attribute k in raw material i
P_{jk}^L	Lower limit of quality attribute k in end product j
P_{jk}^U	Upper limit of quality attribute k in end product j
Decision variables	
q_{il}	Proportion of flow from input stream i to pool l among all flows into pool l
y_{lj}	Flow from pool l to output j
z_{ij}	Flow from input stream i to output j
t_{lj}	Proportion of flow from pool l to output j among all flows out of pool l
x_{il}	Flow from input stream i to pool l

STP-formulation:

$$\min_{\substack{q_{il},y_{lj},\\ z_{i,j}}} \sum_{\substack{(i,l)\in T_X\\ (l,j)\in T_Y}} c_{il}q_{il}y_{lj} - \sum_{(l,j)\in T_Y} d_j y_{lj} - \sum_{(i,j)\in T_Z} (d_j - c_{ij})z_{ij} \tag{B.1}$$

s.t.

$$A_i^L \leq \sum_{\substack{l:(i,l)\in T_X\\ (l,j)\in T_Y}} q_{il}y_{lj} + \sum_{j:(i,j)\in T_Z} z_{ij} \leq A_i^U \qquad \forall i \tag{B.2}$$

$$\sum_{j:(l,j)\in T_Y} y_{lj} \leq S_l \qquad \forall l \tag{B.3}$$

$$D_j^L \leq \sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \leq D_j^U \qquad \forall j \tag{B.4}$$

$$\sum_{\substack{l:(l,j)\in T_Y\\ i:(i,l)\in T_X}} C_{ik}q_{il}y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}z_{ij} \leq P_{jk}^U \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{B.5}$$

$$\sum_{\substack{l:(l,j)\in T_Y\\ i:(i,l)\in T_X}} C_{ik}q_{il}y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}z_{ij} \geq P_{jk}^L \left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \right) \qquad \forall j,k \tag{B.6}$$

$$\sum_{i:(i,l)\in T_X} q_{il} = 1 \qquad \forall l \tag{B.7}$$

$$\sum_{i:(i,l)\in T_X} q_{il}y_{lj} = y_{lj} \qquad \forall l,j \tag{B.8}$$

$$\sum_{j:(l,j)\in T_Y} q_{il}y_{lj} \leq q_{il}S_l \qquad \forall i,l \tag{B.9}$$

$$\sum_{j:(l,j)\in T_Y} t_{lj} = 1 \qquad \forall l \tag{B.10}$$

$$\sum_{j:(l,j)\in T_Y} t_{lj}x_{il} = x_{il} \qquad \forall i,l \tag{B.11}$$

$$\sum_{i:(i,l)\in T_X} t_{lj}x_{il} \leq t_{lj}S_l \qquad \forall l,j \tag{B.12}$$

$$q_{il} y_{lj} = t_{lj} x_{il} \qquad \forall l, i : (i,l) \in T_X, j : (l,j) \in T_Y \tag{B.13}$$

$$0 \leq q_{il} \leq 1 \qquad \forall (i,l) \in T_X \tag{B.14}$$

$$0 \leq y_{lj} \leq \min \left\{ S_l, D_j^U, \sum_{i:(i,l) \in T_X} A_i^U \right\} \qquad \forall (l,j) \in T_Y \tag{B.15}$$

$$0 \leq z_{ij} \leq \min \left\{ A_i^U, D_j^U \right\} \qquad \forall (i,j) \in T_Z \tag{B.16}$$

$$0 \leq t_{lj} \leq 1 \qquad \forall (l,j) \in T_Y \tag{B.17}$$

$$0 \leq x_{il} \leq \min \left\{ S_l, A_i^U, \sum_{j:(l,j) \in T_Y} D_j^U \right\} \qquad \forall (i,l) \in T_X \tag{B.18}$$

REFERENCES

Aarts, E., J. Korst, and W. Michiels, 2005, "Simulated Annealing", In: E. K. Burke, & K. Graham (Eds.), *Search methodologies*, pp. 187-210.

Adhya, N., M. Tawarmalani, and N. V. Sahinidis, 1999, "A Lagrangian Approach to the Pooling Problem", *Industrial & Engineering Chemistry Research*, Vol. 38, No. 5, pp. 1956-1972.

Alfaki, M., 2012, *Models and Solution Methods for the Pooling Problem*, Ph.D. Thesis, The University of Bergen.

Alfaki, M., and D. Haugland, 2012, "Strong Formulations for the Pooling Problem", *Journal of Global Optimization*, pp. 1-20.

Ali, M., A. Törn, and S. Viitanen, 2002, "A Direct Search Variant of the Simulated Annealing Algorithm for Optimization Involving Continuous Variables", *Computers & Operations Research*, Vol. 29, No. 1, pp. 87-102.

Alfi, A., 2011, "PSO with Adaptive Mutation and Inertia Weight and Its Application In Parameter Estimation of Dynamic Systems", *Acta Automatica Sinica*, Vol. 37, No. 5, pp. 541-549.

Al-Khayyal, F. A., 1992, "Generalized Bilinear Programming: Part I. Models, Applications and Linear Programming Relaxation", *European Journal of Operational Research*, Vol. 60, No. 3, pp. 306-314.

Almutairi, H., and S. Elhedhli, 2009, "A New Lagrangean Approach to the Pooling Problem", *Journal of Global Optimization*, Vol. 45(2), pp. 237-257.

Audet, C., J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović, 2004, "Pooling Problem: Alternate Formulations and Solution Methods", *Management Science*, Vol. 50, No. 6, pp. 761-776.

Ben-Tal, A., G. Eiger, and V. Gershovitz, 1994, "Global Minimization by Reducing the Duality Gap", *Mathematical Programming*, Vol. 63, No. 1-3, pp. 193-212.

Bohachevsky, I. O., M. E. Johnson, and M. L. Stein, 1986, "Generalized Simulated Annealing for Function Optimization", *Technometrics*, Vol. 28, No. 3, pp. 209-217.

Byrd, R., J. Nocedal, and R. Waltz, 2006, "Knitro: An Integrated Package for Nonlinear Optimization", In: G. Pillo, and M. Roma (Eds.), *Large-Scale Nonlinear Optimization.*

Cardoso, M. F., R. L. Salcedo, and S. F. de Azevedo, 1994, "Nonequilibrium Simulated Annealing: A Faster Approach to Combinatorial Minimization", *Industrial & Engineering Chemistry Research*, Vol. 33, No. 8, pp. 1908-1918.

Carlisle, A. and G. Dozier, 2001, "An Off-the-shelf PSO", In: *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, USA, 2001.

Černỳ, V., 1985, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications*, Vol. 45, No. 1, pp. 41-51.

Chatterjee, A., and P. Siarry, 2006, "Nonlinear Inertia Weight Variation for Dynamic Adaptation In Particle Swarm Optimization", *Computers & Operations Research*, Vol. 33, No. 3, pp. 859-871.

Clerc, M., 1999, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", In: Evolutionary Computation, 1999. CEC 99. *Proceedings of the 1999 Congress on.*

Clerc, M., and J. Kennedy, 2002, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space", *Evolutionary Computation, IEEE Transactions On*, Vol. 6, No. 1, pp. 58-73.

Corana, A., M. Marchesi, C. Martini, and S. Ridella, 1987, "Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm", *ACM Transactions On Mathematical Software (TOMS)*, Vol. 13, No. 3, pp. 262-280.

Dekkers, A., and E. Aarts, 1991, "Global Optimization and Simulated Annealing", *Mathematical Programming*, Vol. 50. No. 1-3, pp. 367-393.

Dowsland, A. K., 1993, “Simulated Annealing”, In: C. R. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, Inc.

Eberhart, R. C., and J. Kennedy, 1995, “A New Optimizer Using Particle Swarm Theory”, In: Micro Machine and Human Science, MHS’95., *Proceedings of the Sixth International Symposium on.*

Eberhart, R. C., and Y. Shi, 1998, “Comparison Between Genetic Algorithms and Particle Swarm Optimization”, In: *Evolutionary Programming VII.*

Eberhart, R. C., and Y. Shi, 2000, “Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization”, In: Evolutionary Computation, *Proceedings of the 2000 Congress on.*

Eberhart, R.C., and Y. Shi, 2001, “Particle Swarm Optimization: Developments, Applications and Resources”, In: Evolutionary Computation, *Proceedings of the 2001 Congress on.*

Floudas, C. A., and A. Aggarwal, 1990, “A Decomposition Strategy for Global Optimum Search in the Pooling Problem”, *ORSA Journal On Computing*, Vol. 2, No. 3, pp. 225-235.

Foulds, L. R., D. Haugland, and K. Jörnsten, 1992, “A Bilinear Approach to the Pooling Problem”, *Optimization*, Vol. 24, No. 1-2, pp. 165-180.

Frimannslund, L., M. El Ghami, M. Alfaki, and D. Haugland, 2010, “Solving the Pooling Problem with LMI Relaxations”, In: S. Cafieri, B. G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds.), *Proceedings of the Toulouse Global Optimization Workshop*, Tolouse, France, September 2010.

Gounaris, C. E., R. Misener, and C. A. Floudas, 2009, “Computational Comparison of Piecewise-Linear Relaxations for Pooling Problems”, *Industrial & Engineering Chemistry Research*, Vol. 48, No. 12, pp. 5742-5766.

Greenberg, H. J., 1995, “Analyzing the Pooling Problem”, *ORSA Journal On Computing*, Vol. 7, No. 2, pp. 205-217.

Gupte, A., S. Ahmed, S. S. Dey, and M. S. Cheon, 2012, “Pooling Problem”.

Haverly, C., 1978, “Studies of the Behavior of Recursion for the Pooling Problem”, *ACM SIGMAP Bulletin*, Vol. 25, pp. 19-28.

Henderson, D., S. Jacobson, and A. Johnson, 2003, “The Theory and Practice of Simulated Annealing”, In: F. Glover, and G. Kochenberger (Eds.), *Handbook of Metaheuristics.*

Hollander, M., and D. A. Wolfe, 1999, *Nonparametric Statistical Methods*, Second Edition, John Wiley & Sons, New York.

Hu, X., and R. Eberhart, 2002, “Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization”, In: N. Callaos (Ed.) *Proceedings of the sixth world multiconference on systemics, cybernetics and informatics.*

Jiao, B., Z. Lian, and X. Gu, 2008, “A Dynamic Inertia Weight Particle Swarm Optimization Algorithm”, *Chaos, Solitons & Fractals*, Vol. 37, No. 3, pp. 698-705.

Kennedy, J., and R. Eberhart, 1995, “Particle Swarm Optimization”, In: *Proceedings IEEE International Conference on Neural Networks.*

Kennedy, J., and R. C. Eberhart, 1997, “A Discrete Binary Version of the Particle Swarm Algorithm”, In: Systems, Man, and Cybernetics, *Computational Cybernetics and Simulation., 1997 IEEE International Conference on.*

Kennedy, J., and R. Mendes, 2002, “Population Structure and Particle Swarm Performance”, In: Evolutionary Computation, *CEC’02. Proceedings of the 2002 Congress on.*

Kennedy, J., and R. Mendes, 2003, “Neighborhood Topologies in Fully-Informed and Best-of-Neighborhood Particle Swarms”, In: Soft Computing in Industrial Applications, *SMCia/03. Proceedings of the 2003 IEEE International Workshop on.*

Kentzoglanakis, K., and M. Poole, 2009, “Particle Swarm Optimization with an Oscillating Inertia Weight”, In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation.*

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, 1983, “Optimization by Simulated Annealing”, *Science*, Vol. 220, No. 4598, pp. 671-680.

Lasdon, L., A. Waren, S. Sarkar, and F. Palacios, 1979, “Solving the Pooling Problem Using Generalized Reduced Gradient and Successive Linear Programming Algorithms”, *ACM Sigmap Bulletin*, Vol. 27, pp. 9-15.

Liang, J., and P. N. Suganthan, 2005, “Dynamic Multi-Swarm Particle Swarm Optimizer”, In: Swarm Intelligence Symposium, *SIS 2005. Proceedings.*

Liberti, L., and C. C. Pantelides, 2006, “An Exact Reformulation Algorithm for Large Nonconvex NLPs Involving Bilinear Terms”, *Journal of Global Optimization*, Vol. 36, No. 2, pp. 161-189.

Mendes, R., J. Kennedy, and J. Neves, 2004, “The Fully Informed Particle Swarm: Simpler, Maybe Better”, *Evolutionary Computation, IEEE Transactions On*, Vol. 8, No. 3, pp. 204-210.

Misener, R., and C. A. Floudas, 2009, “Advances for the Pooling Problem: Modeling, Global Optimization, and Computational Studies”, *Applied and Computational Mathematics*, Vol. 8, No. 1, pp. 3-22.

Parsopoulos, K. E., and M. N. Vrahatis, 2002, “Particle Swarm Optimization Method for Constrained Optimization Problems”, *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies*, Vol. 76, pp. 214-220.

Peram, T., K. Veeramachaneni, and C. K. Mohan, 2003, “Fitness-Distance-Ratio Based Particle Swarm Optimization”, In: Swarm Intelligence Symposium, *SIS’03. Proceedings of the.*

Poli, R., J. Kennedy, and T. Blackwell, 2007, “Particle Swarm Optimization”, *Swarm Intelligence*, Vol. 1, No. 1, pp. 33-57.

Pulido, G. T., and C. C. Coello, 2004, “A Constraint-Handling Mechanism for Particle Swarm Optimization”, In: Evolutionary Computation, *CEC2004. Congress on.*

Quesada, I., and I. E. Grossmann, 1995, “Global Optimization of Bilinear Process Networks with Multicomponent Flows”, *Computers & Chemical Engineering*, Vol. 19, No. 12, pp. 1219-1242.

Romeijn, H. E., and R. L. Smith, 1994, “Simulated Annealing for Constrained Global Optimization”, *Journal of Global Optimization*, Vol. 5, No. 2, pp. 101-126.

Romeo, F., and A. Sangiovanni-Vincentelli, 1991, “A Theoretical Framework for Simulated Annealing”, *Algorithmica*, Vol. 6, No. 1-6, pp. 302-345.

Schutte, J. F., and A. A. Groenwold, 2005, “A Study of Global Optimization Using Particle Swarms”, *Journal of Global Optimization*, Vol. 31, No. 1, pp. 93-108.

Shi, Y., and R. C. Eberhart, 1998a, “Parameter Selection in Particle Swarm Optimization”, In: V. Porto, N. Saravanan, D. Waagen, and A. Eiben (Eds.), *Evolutionary Programming VII.*

Shi, Y., and R. C. Eberhart, 1998b, “A Modified Particle Swarm Optimizer”, In: Evolutionary Computation Proceedings, *IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on.*

Shi, Y., and R. C. Eberhart, 1999, “Empirical Study of Particle Swarm Optimization”, In: Evolutionary Computation, *CEC 99. Proceedings of the 1999 Congress on.*

Suganthan, P. N., 1999, “Particle Swarm Optimiser with Neighbourhood Operator”, In: Evolutionary Computation, *CEC 99. Proceedings of the 1999 Congress on.*

Tawarmalani, M., and N. V. Sahinidis, 2002, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications.*

Tawarmalani, M., and N. V. Sahinidis, 2005, “A Polyhedral Branch-and-Cut Approach to Global Optimization”, *Mathematical Programming*, Vol. 103, No. 2, pp. 225-249.

Trelea, I. C., 2003, “The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection”, *Information Processing Letters*, Vol. 85, No. 6, pp. 317-325.

Visweswaran, V., and C. Floudast, 1990, "A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs - II. Application of Theory and Test Problems", *Computers & Chemical Engineering*, Vol. 14, No. 12, pp. 1419-1434.

Wächter, A., and L. T. Biegler, 2006, "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming", *Mathematical Programming*, Vol. 106, No. 1, pp. 25-57.

Wicaksono, D. S., and I. Karimi, 2008, "Piecewise MILP Under and Overestimators for Global Optimization of Bilinear Programs", *AIChE Journal*, Vol. 54, No. 4, pp. 991-1008.

Yang, X., J. Yuan, J. Yuan, and H. Mao, 2007, "A Modified Particle Swarm Optimizer with Dynamic Adaptation", *Applied Mathematics and Computation*, Vol. 189, No. 2, pp. 1205-1213.

Li-Ping, Z., Y. Huan-Jun, and H. Shang-Xu, 2005, "Optimal Choice of Parameters for Particle Swarm Optimization", *Journal of Zhejiang University Science A*, Vol. 6, No. 6, pp. 528-534.

Zheng, Y. L., L. H. Ma, L. Y. Zhang, and J. X. Qian, 2003, "On the Convergence Analysis and Parameter Selection in Particle Swarm Optimization", In: Machine Learning and Cybernetics, *International Conference on.*

Zomaya, A. Y., and R. Kazman, 2010, "Simulated Annealing Techniques", In: M. J. Atallah, M. Blanton (Eds.) *Algorithms and theory of computation handbook*, Chapman & Hall/CRC.