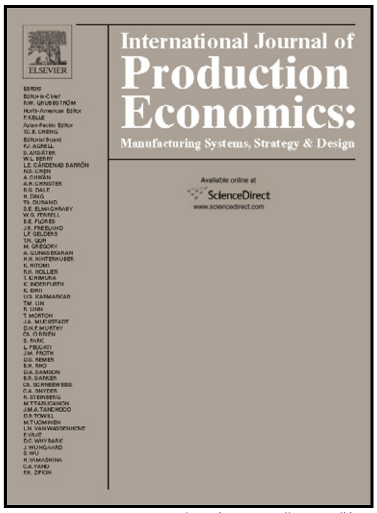


A swarm intelligence based sample average approximation algorithm for the capacitated reliable facility location problem

Nezir Aydin, Alper Murat

PII: S0925-5273(12)00460-4
DOI: http://dx.doi.org/10.1016/j.ijpe.2012.10.019
Reference: PROECO5262

To appear in: *Int. J. Production Economics*

Received date: 22 February 2012
Accepted date: 24 October 2012

A Swarm Intelligence Based Sample Average Approximation Algorithm for Capacitated Reliable Facility Location Problem

Abstract

We present a novel hybrid method, swarm intelligence based sample average approximation (SIBSAA), for solving the capacitated reliable facility location problem (CRFLP). The CRFLP extends the well-known capacitated fixed-cost facility problem by accounting for the unreliability of facilities. The standard SAA procedure, while effectively used in many applications, can lead to poor solution quality if the selected sample sizes are not sufficiently large. With larger sample sizes, however, the SAA method is not practical due to the significant computational effort required. The proposed SIBSAA method addresses this limitation by using smaller samples and repetitively applying the SAA method while injecting social learning in the solution process inspired by the swarm intelligence of particle swarm optimization. We report on experimental study results showing that the SIBSAA improves the computational efficiency significantly while attaining same or better solution quality than the SAA method.

Keywords: reliable, facility location, stochastic programming, sample average approximation, swarm intelligence

1. Introduction

Facility location and strategic supply chain decisions require significant investment and planning for uncertain future events. One example of an uncertain event is the disruption of critical facilities (Schütz, 2009), which can be due to natural disaster or be man-made (i.e., terrorist attacks, labor strikes, etc.). In certain cases, a regional disruption may effect or migrate through other parts of the supply chain (Masihtehrani, 2011). Recent examples of such disruptions include the 2011 earthquake in Japan which affected Toyota's ability to ship parts and finish vehicles (The Guardian, 2011; Brennan, 2011), hurricanes Katrina and Rita in 2005 which disrupted the nation's oil refineries, and the 2000 fire at the Royal Philips Electronics radio frequency chip manufacturing plant in Albuquerque which halted the production of Ericsson and Nokia products (Snyder et al., 2006).

Following a disruption event, it is difficult to rapidly change the supply chain substructure (Snyder et al., 2006). A common recourse is to reassign customers to other facilities or arrange alternative sources of supply. In either case, the cost of serving the customer demand increases due to issues such as higher transportation costs. The effect such disruptions have on supply chain network design has received significant attention in the literature over the past decade. In Snyder and Daskin's exemplary study (2005), a reliability-based formulation for Uncapacitated Facility Location Problem (UFLP) was developed and the authors also proposed a Lagrangean relaxation based algorithm. More recently, Shen et al. (2011) studied a variant of the reliable UFLP in Snyder and Daskin (2005) called uncapacitated reliable facility location problem (URFLP). Shen et al. (2011) proposed highly efficient approximation algorithms for URFLP by exploiting the problem's special structure. However, these approximations cannot be applied to the general class of facility location problems such as capacitated reliable facility location problems (CRFLP).

In practice, capacity and location decisions are jointly considered. Further, facility capacity often cannot be changed in the event of a disruption. Following a facility failure, customers can be assigned to other facilities only if these facilities have sufficient available capacity. Thus, capacitated reliable facility location problems are more complex than their uncapacitated counterparts (Shen et al., 2011). The studies considering capacitated reliable facility location problem are limited. Snyder and Ulker (2005) study the CRFLP and propose an algorithm based on Sample Average Approximation (SAA) embedded with Lagrangean relaxation. Gade (2007) applied the SAA method in combination with a dual decomposition method to solve CRFLP. Peng et al. (2011) proposed a hybrid metaheuristic based on genetic algorithm to solve a related problem where the objective is to minimize the total fixed and transportation cost while limiting the disruption risk based on the p-robustness criterion. In summary, the earlier work on CRFLP uses either SAA based approximation or metaheuristic methods to overcome the computational complexity associated with a large number of scenario realizations.

In this study, we develop a novel technique, called Swarm Intelligence Based Sample Average Approximation (SIBSAA), hybridizing the swarm intelligence and SAA method to efficiently solve the CRFLP. While the standard SAA procedure is effective with sufficiently large samples, the required sample size can be quite large for the desired confidence level. Further, the SAA procedure selects the best performing sample solution and discards the remaining sample solutions which contain valuable information about the problem's uncertainty. The proposed hybrid method reuses the embedded information in sample solutions by iteratively solving the sample problems while injecting social learning in the solution process. The swarm intelligence injection is based on the Particle Swarm Optimization (PSO). Our experimental results indicate that the proposed hybrid method significantly improves computational efficiency while attaining the same or better solution quality as the SAA method.

The rest of this paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we present the capacitated reliable facility location problem and its formulation, briefly summarize the SAA and PSO methods, and then describe the proposed hybrid algorithm in detail. In Section 4, we report on the computational experiments comparing the solution quality and CPU time efficiency of the SAA algorithm and the proposed hybrid algorithm. We conclude with discussion and future research directions in Section 5.

2. Literature Review

Developing efficient methods for solving large-scale stochastic problems is an optimization challenge. There are several exact methods such as Progressive Hedging Algorithm (Rockafeller and Wets, 1991) and L-shaped Decomposition (Van Slyke and Wets, 1969) for solving stochastic programming problems. However, these exact methods become impractical when the number of scenarios is very large. Therefore, we herein focus on two types of approximate methods: sampling based methods and metaheuristics. Sampling methods can be applied in either interior or exterior mode (Linderoth et al., 2006). In the interior mode, the algorithm aims to solve the full problem and only select a sample when an approximate value is needed (Higle and Sen, 1991). In the exterior mode, a sample is randomly selected among all possible scenarios and an approximation of the objective value for the true problem is determined by solving the sample. The SAA method uses exterior sampling and has become a popular technique in solving large-scale stochastic programming problems. This is primarily due to its ease of application. It has been shown that the solutions obtained by the SAA converge to the optimal solution when the sample size is sufficiently large (Ahmed and Shapiro, 2002). However these sample sizes could be quite large and the actual rate of convergence depends on the problem conditioning. Several studies reported successful application of SAA to various stochastic programs (Verweij et al., 2002; Kleywegt et al., 2001; Shapiro and Homem-de-Mello, 1998). The SAA approach is also extensively used in solving stochastic facility location and network design problems (Snyder and Ulker, 2002; Gade, 2007; Santoso et al., 2005; Schütz et al., 2009; Chouinard et al., 2008).

Metaheuristic methods such as Genetic Algorithms (GA), Tabu Search (TS), and Simulated Annealing (SA) have been used to solve stochastic programming problems as an alternative to sampling methods. Kratica et al. (2001) applied GA to solve simple facility location problems. Wang et al. (2008) proposed a stochastic programming-based genetic algorithm to determine a profitable capacity planning and task allocation plan for a resource portfolio planning problem. They reported that using a stochastic sampling procedure improves the effectiveness of the genetic algorithm. Arostegui et al. (2006) compared the TS, SA, GA methods' performances by solving various facility location problems under time-limited, solution-limited, and unrestricted conditions. They reported that the TS method gives better performance overall than both the SA and the GA methods. In this study, we integrate swarm intelligence within the SAA algorithm in an effort to reduce the need to increase sample sizes for improved solution quality. The proposed hybrid method utilizes a swarm intelligence concept similar to that of PSO and enables the swarm learning to take place between the sample solutions of the SAA (i.e., swarm). This combined approach differs from the previous studies using metaheuristics to solve stochastic programming problems in two ways. First, the solution methodology is based on SAA where the solutions in the swarm are obtained by exact solution of the sample subproblems (e.g., rather than an update mechanism such as random crossover operation in GA and velocity update in PSO). Second, the swarm learning is incorporated into the objective function by penalizing deviations from a balanced solution combining the swarm's best solution found thus far and the swarm's average solution. Hence, the swarm learning is an integral part of the solution generation process in the proposed SIBSAA method.

3. Problem Statement and Methodology

In this section, we first present Capacitated Reliable Facility Location Problem (CRFLP) formulation. Next, we summarize the standard SAA and PSO heuristic methods and describe the proposed hybrid SIBSAA methodology in detail.

*3.1. **Capacitated Reliable Facility Location Problem(CRFLP)***

We now introduce the notation used throughout this paper. Let D denote the set of customers (i.e., demand points) and F denote the set of possible facility sites. The fixed cost for facility $i \in F$ is denoted by f_i and incurred if the facility is opened. Let d_j be the demand for customer $j \in D$ and c_{ij} denote the cost of satisfying each unit demand of customer j from facility i and include such variable cost drivers as transportation and production costs. Each facility i has a limited capacity and can serve at most b_i units of demand. Facilities are subject to failure and may be unavailable in the event of a disruption. A customer j cannot be served by any of the facilities whenever all facilities fail, there is not sufficient capacity among the surviving facilities to meet customer j's demand, or the cost of serving customer j's demand via surviving facilities is prohibitive. In such cases, the customer j is assigned to an emergency facility and a large penalty cost is incurred for each unit of unsatisfied demand. The emergency facility can represent an alternative supply source and the large penalty cost then represents the outsourcing cost. In the absence of an alternative supply, the emergency facility corresponds to the lost-sale with the penalty cost of forfeited profit. For simplicity, we denote the last facility in F as the emergency facility and the cost $c_{|F|j}$ as the customer j's unit demand penalty cost.

We formulate the CRFLP as a two-stage stochastic programming problem. In the first stage, the location decisions are made before random failures of the located facilities. In the second stage, following the facility failures, the customer-facility assignment decisions are made for every customer given the facilities that have survived. The goal is to identify the set of facilities to be opened while minimizing the total cost of open facilities and the expected cost of meeting customer demand from the surviving facilities and the emergency facility. In the scenario-based formulation of CRFLP, let s denote a failure scenario and the set of all failure scenarios is S, where $s \in S$. Let p_s be the probability that scenario s occurs and $\sum_{s \in S} p_s = 1$. Further let k_i^s denote whether the facility i survives (i.e., $k_i^s = 1$) and $k_i^s = 0$ otherwise. For instance, in case of independent facility failures, we have $|S| = 2^{|F|-1}$ possible failure scenarios for $|F| - 1$ facilities and the last facility is the emergency facility which is perfectly reliable. Note that our proposed method does not require any assumption on independence and distribution of each facility's failure. To illustrate the CRFLP, let us consider a stylized problem with 10 customers and 5 candidate facilities (Figure 1a). Further, consider a first-stage solution which is to open facilities F_1, F_2, and F_4. Figure 1b illustrates the optimal allocation of demand when all three opened facilities survive. Figure 1c illustrates the optimal allocation of demand when F_4 fails. We note that the customer 2 is assigned to facility F_1 and customer 4 is allocated to F_2. This is because the capacity of facility F_2 is limited to serve only three customers and re-allocating customer 2 allows F_2 to serve the demand of customer 4. Customers 8, 9, and 10 are allocated to the emergency facility (enclosed in dashed circles).

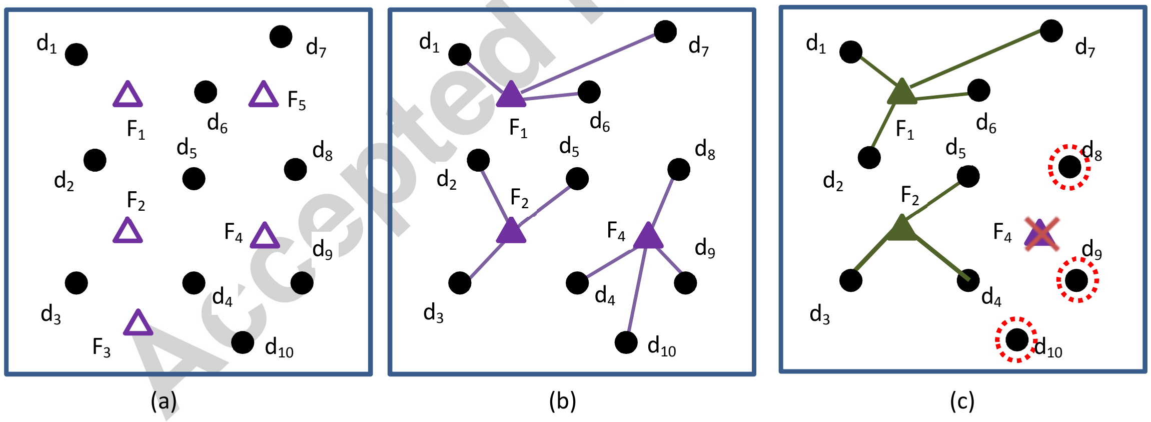


Figure 1. An illustrative example of CRFLP's first-stage and second-stage decisions.

The binary decision variable x_i specifies whether the facility i is opened or not, and the binary variable y_{ij}^s specifies whether the customer j assigned to facility i in scenario s or not. We note that the single sourcing assumption, while a preferred method in practice, is not restrictive for the proposed method. The scenario-based formulation of the CRFLP as a two-stage stochastic program is as follows.

CRFLP:

$$\text{Minimize} \qquad \sum_{i\in F} f_i x_i + \sum_{s\in S} p_s \sum_{j\in D}\sum_{i\in F} d_j c_{ij} y_{ij}^s \tag{1}$$

Subject to

$$\sum_{i\in F} y_{ij}^s = 1 \qquad \forall j\in D, s\in S \tag{2}$$

$$y_{ij}^s \le x_i \qquad \forall j\in D,\ i\in F,\ s\in S \tag{3}$$

$$\sum_{j\in D} d_j y_{ij}^s \le k_i^s b_i \qquad \forall i\in F, s\in S \tag{4}$$

$$x_i, y_{ij}^s \in \{0,1\} \qquad \forall j\in D,\ i\in F,\ s\in S \tag{5}$$

The objective function in (1) minimizes the total fixed cost of opening facilities and the expected second stage cost of satisfying customer demand through surviving and emergency facility. Constraints (2) ensure that each customer is assigned to either an open facility or the emergency facility in every failure scenario. Constraints (3) ensure that a customer's demand cannot be served from a facility that is not opened in every failure scenario. Constraints (4) prevent the assignment of any customer to a facility if it has failed and also to ensure the total demand assigned the facility does not exceed its capacity in every failure scenario. Constraints (5) are integrality constraints.

As stated in the literature review, the main disadvantage of scenario-based formulations is that the number of scenarios grows exponentially with number of facilities. Since enumerating all scenarios is not practical, approximation algorithms are often employed (Snyder et al., 2006). In the next subsection, we discuss two approximation methods (SAA and PSO) and describe the hybrid SIBSAA methodology.

3.2. ***Swarm Intelligence based SAA Method***

The hybrid SIBSAA algorithm integrates a swarm intelligence strategy inspired by PSO with the classical SAA method to improve the solution quality and efficiency. To be thorough, we briefly review the SAA and PSO algorithms in the next two subjections.

3.2.1. ***Sample Average Approximation (SAA)***

The SAA method is commonly used method to solve large-scale stochastic optimization problems (Ahmed and Shapiro, 2002; Kleywegt et al., 2001; Schütz et al., 2009). The main idea of the SAA method is to approximate the objective function value of the stochastic program by solving the problem for a sample of scenarios. After solving the problem for a sufficient number of samples, the first-stage solutions of each sample is tested in a large sample by solving for only the second-stage decisions. The best performing sample solution is then selected as the final solution. In our computational experiments, we compare the performance of the proposed hybrid method with that of SAA. The steps of the SAA procedure to solve the CRFLP are as follows.

SAA Procedure for CRFLP

Initialize: Generate M independent random samples $m = 1, 2, \ldots, M$ with scenario sets N_m where $|N_m| = N$. Each sample m consists of N realizations of independently and identically distributed (i.i.d.) random scenarios. We also select a reference sample which is sufficiently large, e.g., $|N'| \gg N$.

Step 1: For each sample m, solve the following optimization problem and record the sample optimal objective function value v^m and the sample optimal solution $\mathbf{x^m}$=$\{x_i^m\}_{\forall i \in F}$.

<u>SAA-CRFLP (m):</u>

$$\text{Minimize} \qquad \sum_{i \in F} f_i x_i^m + \sum_{s \in N_m} \frac{1}{|N_m|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{sm} \tag{6}$$

Subject to

$$\sum_{i \in F} y_{ij}^{sm} = 1 \qquad \forall j \in D, s \in N_m \tag{7}$$

$$y_{ij}^{sm} \leq x_i^m \qquad \forall j \in D,\ i \in F,\ s \in N_m \tag{8}$$

$$\sum_{j \in D} d_j y_{ij}^{sm} \leq k_i^s b_i \qquad \forall i \in F, s \in N_m \tag{9}$$

$$x_i^m, y_{ij}^{sm} \in \{0,1\} \qquad \forall j \in D,\ i \in F, s \in N_m \tag{10}$$

Step 2: Calculate the average $\overline{v}^M$ of the sample optimal objective function values obtained in Step 1 as follows.

$$\overline{v}^M = \frac{1}{M} \sum_{m=1}^{M} v^m \tag{11}$$

Step 3: Estimate the true objective function value $\hat{v}^m$ of the original problem for each sample's optimal solution. Solve the following optimization problem for each sample using the optimal first stage decisions $\mathbf{x^m}$ from Step 1.

$$\hat{v}^m = \text{Minimize} \sum_{i \in F} f_i x_i^m + \sum_{s=1}^{|N'|} \frac{1}{|N'|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{sm} \tag{12}$$

Subject to constraint sets (2), (3), (4), (5), and using $N_m \equiv N'$.
Step 4: Select the solution $\mathbf{x^m}$ with the best $\hat{v}^m$, i.e. $\mathbf{x^{S}}^{AA} = argmin_{m=1,\ldots,M}\ \hat{v}^m$ as the solution and $v^{SAA} = \min_{m=1,\ldots,M} \hat{v}^m$, as the solution value of SAA.

Let v^* denote the optimal objective function value of the original problem CRFLP. The $\overline{v}^M$ is an unbiased estimator of $\mathbb{E}[v]$ which is the expected optimal objective function value of sample problems. Since $\mathbb{E}[v] \leq v^*$, the $\overline{v}^M$ provides a statistical lower bound on the v^* (Ahmed and Shapiro, 2002). While $\overline{v}^M$ does not always provide a lower bound on the v^*, as observed in Shen et al. (2011), it is useful in assessing the quality of the solution value of SAA $\hat{v}^{SAA}$. The reference set N' is used to estimate the objective function value of the sample problem solutions in the original problem CRFLP.

3.2.2. ***Particle Swarm Optimization (PSO)***

The PSO methodology, first proposed by Kennedy and Eberhart (1995), is a population-based metaheuristic search technique motivated by the social behavior of organisms such as bird flocking and fish schooling. It is based on the swarm intelligence idea where knowledge is optimized by social interaction and thinking is not only personal but also social. The PSO achieves local and global search capabilities where the intensification and diversification are achieved via relative weighting of the personal and social (swarm-based) thinking. PSO has been successfully applied in solving many combinatorial optimization problems in which the objective space possesses many local optimal solutions. In such problems where there are many multiple local optima, the PSO provides the advantage of escaping from the local optima through the communication between members of the swarm.

In PSO, each solution is represented as a particle in a swarm. Each particle has a position and a fitness value that is evaluated by the fitness function to be optimized. The particles iterate from one position to another through a velocity vector. This velocity vector is determined by the particle's most recent displacement, its best position thus far, and the best position encountered by all particles in the swarm. The velocity vector of each particle is calculated by updating the previous velocity by two best values: *pbest* and *gbest*. The personal best (*pbest*) is the particle's best position it has visited thus far and tracked by each particle. The global best (*gbest*), tracked by the swarm, is the best position visited by any particle in the population. The influence of the personal and global bests on the velocity vector is controlled by weights called learning factors.

Let L be the set of particles, $l = 1, \ldots, L$ in swarm. In every iteration of the PSO, the particle l's velocity at iteration k, $u_l\,(k)$, is updated according to :

$$u_l\,(k+1) = \gamma u_l\,(k) + \delta_1 r_1\,[\theta_l\,(k) - x_l\,(k)] + \delta_2 r_2\,[\mu\,(k) - x_l\,(t)] \tag{13}$$

where $x_l\,(k)$ is the position (solution) of particle l at iteration k, and the parameters γ and (δ_1, δ_2) are inertia weight and learning factors controlled by the user, respectively. The weights r_1 and r_2 are uniform random numbers generated at every velocity update. The value $\theta_l\,(k)$ is the individual best solution for particle l at iteration k, and $\mu\,(k)$ is the swarm's global best solution at iteration k. Using this velocity, the particle's position is updated as $x_l\,(k+1) = x_l\,(k) \oplus u_l\,(k+1)$ where the operator $\oplus$ represents the update scheme depending on whether x_l is continuous or discrete (Kennedy and Eberhart, 1997).

The three terms in (13) represent the memory, cognitive learning and social learning of the particle, respectively. The $\gamma u_l\,(k)$ is called the particle's inertia which induce the particle to move in the same direction as the most recent displacement. The weights (δ_1, δ_2) are referred as the learning rates since the inertia weight controls the extent to which the memory of the previous velocity influences the new velocity. The diversification and intensification of the particle is controlled through the inertia weight as well as velocity bounds which limit velocity to preset maximum and minimum levels (Shi and Eberhart, 1998). Inertia weight, velocity bounds, and learning rates jointly determine the particle's motion. Usually, a high inertia weight is used at the beginning and then gradually decreased so as to diversify the solution.

3.2.3. ***Hybrid Method: Swarm Intelligence based SAA (SIBSAA)***

The proposed method is a hybridization of the SAA and PSO. The motivation for this hybridization originates from the last stage of the SAA method (Step 4) where the best performing solution is selected and the rest of the solutions are discarded. However, this discarding of $(M-1)$ sample solutions is a loss of valuable sample information as well as a loss of effort spent in solving each sample's solution. Let's consider the implementation of the classical SAA procedure in the context of PSO and treat each sample solution as a particle. Then the implementation of SAA would correspond to iterating the particles (sample solutions) in the swarm (the set of sample solutions) only once and then selecting the best performing particle. In the PSO, however, the particle iterations are sustained with the particle's recent memory (e.g., inertia) as well as with the social learning. Hence, in the proposed hybrid approach, we modify the SAA method by continuing the solution of the sample problems (e.g., particles) while injecting the social learning in the solution process. The underlying premise of this hybridization is that, by starting with sufficient number of samples (representative of the entire scenario set), the continued iteration of the SAA method with social learning would converge the sample solutions to the optimal solution of the original problem.

An important distinction of the proposed hybrid method from the classical PSO is the movement of particles. The PSO iterates the particles according to a velocity vector and the positions of particles are suboptimal solutions (until convergence to the global optimum). In comparison, the SIBSAA solves the sample problems to optimality. However, since the samples do not correspond to the entire scenario set, these optimal solutions are sub-optimal for the original problem. As a result of this distinction, the injection of the social learning in the particle's movement is different in the SIBSAA than that in PSO. The social learning in SIBSAA is achieved through dynamically penalizing the deviation of the sample solution from the swarm's balanced solution ($\overline{\overline{\mathbf{x}}}^k$) at iteration k. This balanced solution is composed of the swarm's best incumbent solution and the swarm's average solution at iteration k. The swarm's best incumbent solution ($\mathbf{x}_{best}$) is the solution with the best objective value ($\hat{v}_{best}$) considering the reference set N'. On the other hand, the swarm's average solution ($\overline{\mathbf{x}}^k$) at iteration k is the probability weighted average of sample solutions at iteration $(k-1)$. Similar to the PSO, social learning is modulated with parameters ρ^k, ω_m^k, and α^k. The parameter ρ^k is the swarm learning parameter which penalizes the sample solutions' squared Euclidean norm deviation from the swarm's balanced solution at iteration k. The parameter ω_m^k penalizes the linear deviation of the sample solutions from the swarm's balanced solution at iteration k. Lastly, the parameter α^k is a balancing weight between the swarm's average solution and the swarm's best incumbent solution, and is used to calculate the swarm's balanced solution.

We first present the proposed SIBSAA algorithm and then describe its steps in detail. For brevity, we use the same notation as before and only introduce the additional notation used in the SIBSAA algorithm.

<u>Notation:</u>

k, k_{max} : iteration index and maximum number of iterations
$P_m, \hat{P}_m$: probability and normalized probability of realization of sample m
$\mathbf{x}^{m,k}$: solution vector for sample m at iteration k
$\overline{\mathbf{x}}^k$: swarm's average sample solution at iteration k
$\overline{\overline{\mathbf{x}}}^k$: swarm's balanced solution at iteration k
$\mathbf{x}_{best}$: best incumbent solution
$\hat{v}_{best}$: objective function value of the best incumbent solution with respect to N'
$\hat{v}^k_{best}$: objective function value of the best solution at iteration k with respect to N'
ω^k_m : dual variable vector for sample m at iteration k
ρ^k : swarm learning parameter at iteration k
β : update factor for the swarm learning parameter
α^k : weight for learning from the global best at iteration k
$\triangle_\alpha$: update parameter for global learning weight
ϵ_k : Euclidean norm distance of sample solutions from the $\overline{\overline{\mathbf{x}}}^k$ at iteration $k-1$
ε : convergence threshold for solution spread

$\mathbf{x}^{SIBSAA}$: best solution found by SIBSAA
v^{SIBSAA}: objective function value of the best solution found by SIBSAA

The pseudo-code for the swarm intelligence based SAA is as follows:

Swarm Intelligence based SAA Algorithm (SIBSAA) for CRFLP

Initialize: Generate M independent random samples $m = 1, 2, \ldots, M$ with scenario sets N_m where $|N_m| = N$. Each sample m consists of N realizations of independently and identically distributed (i.i.d.) random scenarios. We also select a reference sample which is sufficiently large, e.g., $|N'| \gg N$.

- Set $k := 0$, $\omega_m^{k=0} = 0$ for $\forall m = 1, \ldots, M$, select $\alpha^k = 0 \in [0, 1]$, and $\rho^{k=0} := 1$,
- Calculate $P_m := \prod_{s \in N_m} p_s$ (i.i.d. scenarios in sample m) and $\hat{P}_m = \frac{P_m}{\sum_{m=1}^{M} P_m}$ and denote $\widehat{\mathbf{P}} = \left\{ \hat{P}_m \right\}_{\forall m}$.

Step 1: Execute the Steps 2-4 of the SAA Algorithm for CRFLP using the scenario sets N_m for $m = 1, 2, \ldots, M$ and the reference set N'.

1.1. Assign the solutions $\mathbf{x}_{best} := \mathbf{x}^{SAA}$ and $\mathbf{x}^{m,k=0} = \mathbf{x}^m$ for $\forall m = 1, \ldots, M$.

Step 2: Update the problem parameters as follows,

2.1. Set $k := k + 1$,

2.2. Calculate $\overline{\mathbf{x}}^k := \widehat{\mathbf{P}} \mathbf{x}^{m,k-1}$,

2.3. Calculate $\overline{\overline{\mathbf{x}}}^k := \alpha^k \overline{\mathbf{x}}^k + \left(1 - \alpha^k\right) \mathbf{x}_{best}$,

2.4. Update $\alpha^k := \alpha^{k-1} - \triangle_\alpha$,

2.5. Update ρ^k if $k > 2$,

$$\rho^k := \begin{cases} \beta \rho^{k-1} & if \quad \epsilon_k > \epsilon_{k-1}/2 \\ \rho^{k-1} & otherwise \end{cases}$$

2.6. Calculate $\omega_m^k := \omega_m^{k-1} + \rho^k \left(\mathbf{x}^{m,k-1} - \overline{\overline{\mathbf{x}}}^k \right)$.

Step 3: For each sample $m = 1, \ldots, M$, solve the following optimization problem and record the sample optimal objective function value $v^{m,k}$ and the sample optimal solution $\mathbf{x}^{m,k} = \left\{ x_i^{m,k} \right\}_{\forall i \in F}$.

<u>SIBSAA-CRFLP(m):</u>

$$\text{Minimize} \quad \sum_{i \in F} f_i x_i^{m,k} + \sum_{s \in N_m} \frac{1}{|N_m|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{smk} + \omega_m^k \mathbf{x}^{m,k} + \frac{\rho^k}{2} \left\| \mathbf{x}^{m,k} - \overline{\overline{\mathbf{x}}}^k \right\|^2 \tag{14}$$

Subject to constraints (7), (8), (9), and (10). Update $\epsilon_k := \left(\sum_{m=1}^{M} \left\| \mathbf{x}^{m,k} - \overline{\overline{\mathbf{x}}}^k \right\| \right)^{1/2}$.

Step 4: Using the sample solutions $\mathbf{x}^{m,k}$ for $m = 1, \ldots, M$ obtained in Step 3, estimate the true objective function value $\hat{v}^m$ of the original problem by solving the following optimization problem for each sample.

$$\hat{v}^{m,k} = \text{Minimize} \sum_{i \in F} f_i x_i^m + \sum_{s=1}^{|N'|} \frac{1}{|N'|} \sum_{j \in D} \sum_{i \in F} d_j c_{ij} y_{ij}^{smk} \tag{15}$$

Subject to constraints (7), (8), (9), and (10) using $N_m \equiv N'$.

Step 5: Select the solution $\mathbf{x}^{m,k}$ with the best $\hat{v}^{m,k}$. Let $\hat{v}^k_{best} = argmin_{m=1,\dots,M}\ \hat{v}^{m,k}$, then

$$\hat{v}_{best} := \begin{cases} \hat{v}^k_{best} & if\ \ \hat{v}^k_{best} < \hat{v}_{best} \\ \hat{v}_{best} & otherwise \end{cases}$$

$$\mathbf{x}_{best} := \begin{cases} \mathbf{x}^{m',k} | m' = argmin_{m=1,\dots,M}\ \hat{v}^{m,k} & if\ \ \hat{v}^k_{best} < \hat{v}_{best} \\ \mathbf{x}_{best} & otherwise \end{cases}$$

Step 6: Check for convergence: If ($\epsilon_k \geq \varepsilon$ or $\overline{\overline{\mathbf{x}}}^k \neq \mathbf{x}_{best}$) and ($k < k_{max}$), then return to Step 2, otherwise terminate with best found solution $\mathbf{x}^{\mathbf{S}IBSAA}$=$\mathbf{x}_{best}$ and solution value $v^{SIBSAA} = \hat{v}_{best}$.

The initialization step of the SIBSAA is similar to that of SAA's and the only additional calculation is the sample m's probability and normalized probabilities, e.g., P_m and $\hat{P}_m$. The first step in SIBSAA is to execute the standard SAA procedure (Step 1). Next, we calculate the swarm's average solution and the balanced solution. In Step 2.2, we calculate the probability weighted average $\overline{\mathbf{x}}^k$ of swarm's solutions. The swarm's balanced solution ($\overline{\overline{\mathbf{x}}}^k$), calculated in Step 2.3, is a weighted average of the average solution ($\overline{\mathbf{x}}^k$) and the incumbent best solution ($\mathbf{x}_{best}$). The idea of using $\mathbf{x}_{best}$ in SIBSAA is analogous to the use of global best in PSO. However, $\overline{\mathbf{x}}^k$ is unique to SIBSAA and ensures that the swarm's solutions converge to the same solution, i.e., sample average solution. This is because the sample solutions are determined based on the scenario set of each sample, whereas the ultimate goal is to identify the best solution of CRFLP with respect to the reference set N'. The attainment of this goal can be improved if every sample's solution is determined by accounting for the solutions of other samples. Hence, while solving each sample's problem (e.g., determining a particle's position), the SIBSAA trades off between the sample's optimum solution (e.g., particle's individual best), best solution among all samples with respect to the reference set N'(i.e., swarm's global best), and sample average solution ($\overline{\mathbf{x}}^k$). The trade-off between the latter two, i.e., best solution and sample average solution, is achieved through the weight factor $\alpha^k \in [0,1]$ in Step 2.3 which determines the bias of the social learning; whereas high values tend the sample solutions to the sample average solution, low values tend to the incumbent best solution. Note that $\overline{\mathbf{x}}^k$, calculated in Step 2.2, is always in the convex hull of sample solutions (i.e., particles). Further, the penalty term in the objective varies from iteration to iteration as ($\overline{\overline{\mathbf{x}}}^k$) evolves with changing $\mathbf{x}_{best}$ and penalty multiplier ρ. Hence, the convex hull of solutions evolves during the search (e.g., expands/shrinks, changes shape, shifts, and number of vertices increase/decrease–not exceeding number of particles). This evolution therefore allows for the exploration of the search space.

There are two strategies for modulating the social learning bias; α^k can be static by setting $\triangle_\alpha = 0$ or can be dynamically varied between iterations by setting $\triangle_\alpha > 0$ (see Step 2.4). The advantage of dynamic α^k is that, beginning with a large α^k, we first prioritize the sample average solution until the incumbent best solution quality improves. This approach allows guiding the sample solutions to a consensus sample average initially and then directing the consensus sample average in the direction of improved best incumbent solution. The alternative strategy of starting with a low α^k will initially tend the sample solutions to a low quality solution, which in turn is likely to result in low quality sample average solutions in the subsequent iterations.

In Step 2.5, we update the swarm learning parameter ρ^k depending whether the distance (ϵ_k) of sample solutions from the most recent balanced solution has sufficiently improved. We choose the improvement threshold as half of the distance in the previous iteration (e.g., ϵ_{k-1}). Similarly, in Step 2.6, we update the penalty parameter (ω_m^k) for the linear deviation of every sample's solution from the swarm's balanced solution at iteration k. Note that the ω_m^k are the Lagrange multipliers corresponding to the equivalence of each sample's solution to the balanced solution.

In Step 3, we solve each sample problem with additional objective function terms representing the social learning and calculate the deviation of the sample solutions from the balanced solution (i.e., ϵ_k). Step 4 estimates the objective function value of each sample solution in the original problem using the reference set N'. Step 5 identifies the sample solution $\mathbf{x}^{m,k}$ with the best $\hat{v}^{m,k}$ in iteration k and updates the incumbent best $\hat{v}_{best}$ if there is improvement. Steps 4 and 5 correspond to the integration of SAA method's selection of the best performing sample solution. In contrast to SAA's termination with the best performing sample solution, the proposed SIBSAA retains this information to induce social learning in the next iteration through the balanced solution. Step 6 checks whether the stopping conditions are met. If the iteration limit is reached $k \geq k_{max}$ or when all the sample solutions converged to the balanced solution within a tolerance, then the SIBSAA terminates with the best found solution. The worst-case solution of the SIBSAA is equivalent to the SAA solution with the same set of samples. This can be observed by noting that the best incumbent solution is initialized with the SAA's solution. Hence, the SIBSAA converges to a solution which has the same performance or better than that of SAA's.

4. Experimental Study

We now describe the experimental study performed to investigate the computational and solution quality performance of the proposed SIBSAA for solving CRFLP. We benchmark the results of SIBSAA with those of SAA and an exact solution. We solved the CRFLP exactly by using the deterministic equivalent formulation. All algorithms are programmed in Matlab R2010b and integer programs are solved with CPLEX 12.1. The experiments are conducted on a PC with Intel(R) Core 2 CPU, 2.13 GHz processor and 2.0 GB RAM running on Windows 7 OS. Next, we describe the experimental setting and data in detail. In Section 4.2, we report on sensitivity analysis results of SIBSAA's performance with respect to algorithm's parameters. In Section 4.3, we present and discuss the benchmarking results.

4.1. ***Experimental Setting***

We used the test data sets from Zhan (2007) that are also used in Shen et al. (2011) for the URFLP. In these data sets, the coordinates of site locations are i.i.d. and are sampled from $\mathrm{U}[0,1] \times \mathrm{U}[0,1]$. The customer and facility sites are identical. The customer demand is i.i.d, sampled from $\mathrm{U}[0,1000]$, and is rounded to the nearest integer. The fixed cost of opening a facility is i.i.d. and is sampled from $\mathrm{U}[500,1500]$, and is rounded to the nearest integer. The variable costs c_{ij} for $i = 1,\ldots,|F|-1$ and $\forall j$ are chosen as the Euclidean distance between sites. The penalty cost $c_{|F|j}$ for serving customer j from the emergency facility is i.i.d. and is sampled from $\mathrm{U}[0,15]$. Since Zhan (2007) and Shen et al. (2011) consider uncapacitated RFLP, the data sets do not have facility capacities. We have selected identical capacity levels for all facilities $b_{i=1,..,|F|} = 2,000$. In generating the failure scenarios, we assume that the facility failures are independently and identically distributed according to the Bernoulli distribution with probability q_i (i.e., the failure probability of facility i). In our experiments, we use uniform failure probability (i.e., $q_{i=1,\ldots,|F|-1} = q$,) and consider the cases $q = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The emergency facility is perfectly reliable (i.e., $q_{|F|} = 0$). Note that the case $q = 0$ corresponds to the deterministic fixed-charge facility location problem, and $q = 1$ corresponds to the case where all facilities fail. The latter solution is where all customers are assigned to the emergency facility and the objective function value is then $\sum_{j \in D} d_j c_{|F|j}$. The failure scenarios $s \in S$ are generated as follows. Let $F_f^s \subset F$ be the facilities that are failed, and $F_r^s \equiv F \backslash F_f^s$ be the set of facilities that are reliable (i.e., not failed) in scenario s. The facility indicator parameter in scenario s become k_i^s=1 if $i \in F_r^s$, and k_i^s=0 otherwise. The probability of scenario s is then calculated as $p_s = q^{\left|F_f^s\right|}(1-q)^{|F_r^s|}$.

In all experiments, we used $|D| = |F| - 1 = 10$ sites which is a medium-sized CRFLP problem and is more difficult to solve than the uncapacitated version (URFLP). The size of the failure scenario set is $|S|$=1,024. The deterministic equivalent formulation has variables x_i and y_{ij}^s totaling $|F|+|F|\times|D|\times|S| = 11+11\times10\times1024 = 112,651$ binary variables. Similarly, it has constraints (7), (8) and (9) totaling $|D|\times|S|+|F|\times|D|\times|S|+|F|\times|S| = 11+11\times10\times1024 = 134,144$ constraints. We generated sample sets for SAA and the proposed SIBSAA by randomly sampling from U $[0, 1]$ as follows. Given the scenario probabilities, p_s, we calculate the scenario cumulative probability vector $\{p_1, (p_1 + p_2), \ldots, (p_1 + p_2 + \ldots + p_{|S|-1}), 1\}$ which has $|S|$ intervals. We first generate the random number and then select the scenario corresponding to the interval containing the random number. We tested the SAA and SIBSAA algorithms with varying number of samples (M), and sample sizes (N). We used identical sample sets in all the SAA and SIBSAA comparisons where M and N are same. In selecting the reference set, we use the entire scenario set, i.e., $N' = S$, as the evaluation of a solution with $|S|$=1,024 scenarios can be easily computed . All the datasets used in the following sections and results are available from the authors upon request.

4.2. ***SIBSAA Parameter Sensitivity***

We evaluated the sensitivity of the SIBSAA with respect to the social learning bias parameter (α), number of samples (M), and the swarm learning parameter update factor (β). First, we tested for α which determines the bias of the social learning (e.g., biased towards swarm's best or swarm's average sample solution at iteration k). In these tests, we set $(M, N) = (5, 10)$, $q = 0.4$, and $\beta = 2$ unless otherwise is stated. Table 1 the first column shows the type of social learning strategy (i.e., static and dynamic bias). The second column is the parameter of the corresponding strategy (i.e., $\triangle_\alpha$ for dynamic and α for static). Note that in the dynamic strategy, we select initial value as $\alpha^{k=0} = 1$. The first stage solutions and corresponding objective function value is shown in the third and fourth columns. The last column presents the CPU time in terms of seconds which includes solving sample subproblems and SIBSAA's overhead in Step 2. The exact solution is shown in the bottom row; this solution took longer than six hours.

The first observation is that the SIBSAA is relatively insensitive to the strategy employed and the parameter settings. In the dynamic case, $\triangle_\alpha = 0.05$ identifies the optimal solution, and the remainder parameter settings identify a similar solution except the facility 7 is opened in place of facility 4. Further, as the $\triangle_\alpha$ increases, the swarm's best incumbent solution becomes increasingly more important leading to decreased computational time. The static strategy, similarly, identifies the exact solution when $\alpha = 0.6$ and $\alpha = 0.8$ and finds the same near-optimal solution in other settings. The CPU time with the static strategy is variable and slightly higher than the dynamic strategy on the average. In comparison, the dynamic strategy converges to a solution faster than the static strategy (e.g., average of 63.5 versus 69.2 seconds in Table 1, respectively). These results show that the SIBSAA's sensitivity to the α parameter strategy and settings is nominal.

Next, we tested the SIBSAA's sensitivity with respect to the number of samples by varying M from 3 to 20, while keeping the sample size same ($N = 10$). For brevity, we illustrate only the results for the dynamic strategy with $\triangle_\alpha = 0.05$ and the static strategy with $\alpha = 0.8$. The remainder of the settings are taken as before.Figure 2 shows that increasing the number of samples improves the quality of the solution in both static and dynamic strategies. Further, the SIBSAA algorithm is able to converge to the optimal solution even with a small number of samples, e.g., $M = 5$. While the CPU time increases with an increasing number of samples, this increase is linear in M and the CPU time performances of both strategies are similar.

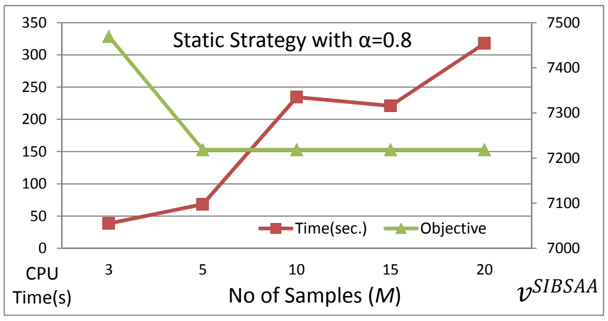
Lastly, we tested the sensitivity of the SIBSAA with respect to the swarm learning update (β) using both the static and dynamic strategies of the swarm learning bias (Figure 3). In the static strategy, increasing the swarm learning rate results in faster convergence but with an inferior solution. In comparison, with the dynamic strategy using $\triangle_\alpha = 0.05$, the solution quality first improves and then declines as the CPU time decreases (Figure 3b). This indicates an interaction between the swarm learning bias parameter and swarm learning update. While the dynamic strategy can converge to a solution faster than the static strategy, the swarm learning update and bias parameters need to be jointly tuned to ensure high solution quality. Lastly, we note that as the bias parameter ($\triangle_\alpha$) and the swarm learning update (β) increase, the convergence rate increases. However, at the same time, the solution quality might degrade as the best incumbent solution might not have sufficiently improved. Hence, a good tuning strategy is to use high (low) β with low (high) $\triangle_\alpha$ in order to achieve fairly speedy convergence to a good quality solution.

α				
Strategy	Parameter	Open Facilities	Objective	Time(sec.)
Dynamic	$\triangle_\alpha$=0.03	1,2,7,8,10	7,338	69.2
	$\triangle_\alpha$=**0.05**	**1,2,4,8,10**	**7,218**	**64.8**
	$\triangle_\alpha$=0.08	1,2,7,8,10	7,338	64.7
	$\triangle_\alpha$=0.10	1,2,7,8,10	7,338	61.1
	$\triangle_\alpha$=0.12	1,2,7,8,10	7,338	62.2
	$\triangle_\alpha$=0.15	1,2,7,8,10	7,338	59.2
Static	α=0.5	1,2,7,8,10	7,338	69.8
	α=**0.6**	**1,2,4,8,10**	**7,218**	**73.6**
	α=0.7	1,2,7,8,10	7,338	61.3
	α=**0.8**	**1,2,4,8,10**	**7,218**	**68.3**
	α=0.9	1,2,7,8,10	7,338	69.8
	α=1	1,2,7,8,10	7,338	72.5
Exact Solution		**1,2,4,8,10**	**7,218**	**$\gg$21,600**

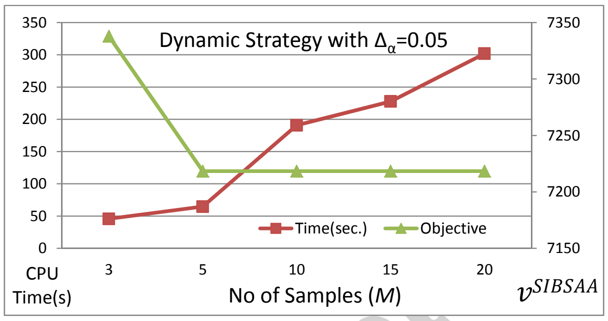
Table 1. Sensitivity of SIBSAA to social learning bias parameter (α) strategy and settings

4.3. Computational Performance of SIBSAA

In this section, we compare the performances of the SAA and the proposed SIBSAA. In all experiments, we use the same settings of β= 2 and static strategy for learning bias with α= 0.8. We analyzed the performance of the proposed SIBSAA with respect to that of the exact method and the SAA method with different sample sizes (N) and number of samples (M). For SAA method, we experimented with M={5,10,20} and N={10,25,50}. In comparison, all SIBSAA sample sets

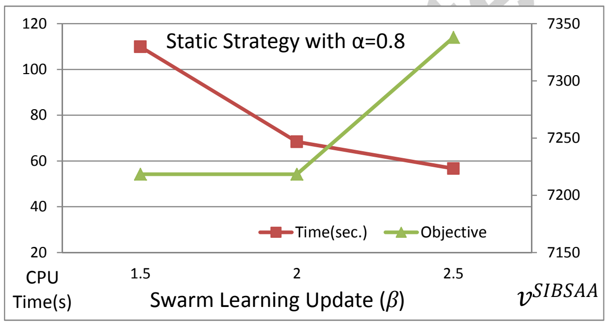


(a)

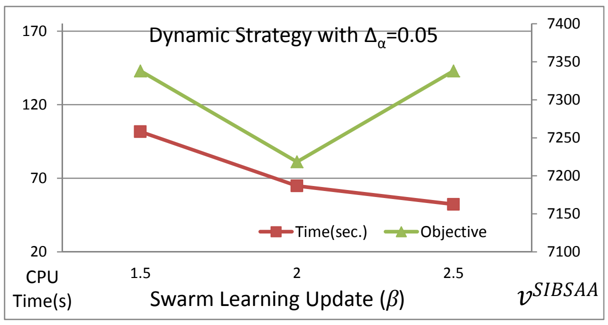


(b)

Figure 2. Effect of number of samples (M) on SIBSAA's performance in the case of (a) static strategy with $\alpha = 0.8$, and, (b) dynamic strategy $\triangle_\alpha = 0.05$



(a)



(b)

Figure 3. Effect of the swarm learning parameter update (β) on SIBSAA's performance in the case of (a) static strategy with $\alpha = 0.8$, and, (b) dynamic strategy $\triangle_\alpha = 0.05$

have M={5} with varying sample size, e.g., N={5,10,15} We report on the results based on 10 replications (e.g., we generated 10 random sample instances for each combination of sample size and number of samples) and solved with SIBSAA and SAA methods. Tables 2 and 3 illustrate these benchmark results for q={0.1,0.3,0.5,0.7}. For each failure probability, we report on the results for best solution as well as various statistics of 10 replications.

The F^*_{Best} column indicates the best solution converged by each method across all replications (e.g., facilities opened). For instance, with $q = 0.3$, the SAA's solution is to open facilities F^*_{Best}={1,2,3,10} with $M = 5$ and $N = 10$, whereas the SIBSAA (for all sample sizes) and exact solution method open facilities F^*_{Best}={1,2,8,10}. The Best Obj. column presents the best objective function value found for the SAA, SIBSAA, and exact method (e.g., v^{SAA}, v^{SIBSAA} and v^*). The $\overline{v}^M$ column is the average of the sample optimal objective functions for the sample set corresponding to the replication with the best objective. We present two optimality gap measures. The first gap (GAP$_1$) is only applicable for the SAA since it is based on the assumption that $\overline{v}^M$ is a statistical lower bound on the v^*. It is defined as,

$$\text{GAP}_1 = \frac{v^{SAA} - \overline{v}^M}{\overline{v}^M} \times 100\%.$$

For each failure probability, we present the worst, average, and standard deviation of the objective function values, average and standard deviation of CPU times, and second type of optimality gap across all replications. The second type of optimality gap (GAP$_2$) is applicable to both the SAA and the SIBSAA, and uses the optimal solution value v^*. It is defined as,

$$\text{GAP}_2 = \begin{cases} \frac{v^{SAA} - v^*}{v^*} \times 100\% & \text{for SAA,} \\ \\ \frac{v^{SIBSAA} - v^*}{v^*} \times 100\% & \text{for SIBSAA.} \end{cases}$$

Tables 2 and 3 show that, as the sample size increases, the SAA's objective function is not always monotonously decreasing while the CPU time is increasing exponentially. Similarly, increasing the sample size increases the SIBSAA's CPU time, but the rate of increase is less than that of the SAA. This is attributable to the reduced number of SIBSAA iterations and increased effectiveness of using warm starts (e.g., using the previous iteration's solution as the initial solution for the current sample problem). However, when the sample size of SIBSAA is increased beyond $N = 15$, the CPU time is observed to increase exponentially (e.g., similar to that of SAA) as the advantages of the warm start and improved convergence have plateaued. While the results in Table 2 and 3 demonstrate that $N = 10$ or $N = 15$ are sufficient for improved solution quality over SAA, it is critical to determine a balanced sample size for the effectiveness of SIBSAA (e.g., where the optimality benefit of increasing the sample size justifies the increased CPU time).

Note that the negative GAP$_1$ values are due to the fact that $\overline{v}^M$ is a "statistical" lower bound. In Table 2, with $q = 0.1$, we observe that the SAA method finds the optimal solution in all M and N cases except $(M, N) = \{(5, 10), (5, 25), (10, 10)\}$ whereas the SIBSAA's solution is optimal except $(M, N) = \{(5, 5)\}$. With this failure probability, the CRFLP is relatively easy to solve since the effect of sampling is not significant.

With $q = 0.3$, the SIBSAA's average GAP$_2$ is about 0.1% with an average CPU time of 159 seconds. In comparison, the SAA's average GAP$_2$ and CPU time are 3.6% and $1,542$ seconds. Across all replications, the SAA finds the optimal solution for samples $(M, N) = \{(10, 50), (20, 50)\}$

whereas the SIBSAA can find optimum solution for samples $(M,N) = \{(5,10),(5,15)\}$ within much less CPU time.

		q=0.1										q=0.3									
		Best Solution				Randomized Trials						Best Solution				Randomized Trials					
Method	M-N	F^*_{Best}	Best Obj.	$\overline{v}^M$	GAP_1 (%)	Ave. Obj.	Worst Obj.	Stdev. Obj.	CPU(s) Ave	CPU(s) Stdev	GAP_2 (%)	F^*_{Best}	Best Obj.	$\overline{v}^M$	GAP_1 (%)	Ave. Obj.	Worst Obj.	Stdev. Obj.	CPU(s) Ave	CPU(s) Stdev	GAP_2 (%)
SAA	5-10	1,2,4	4,322	4,194	3.1	4,442	4,542	97	3	0	2.8	1,2,3,10	6,456	5,723	12.8	6,793	7,375	359	16	2	10.5
	5-25	1,2,10	4,322	4,251	1.7	4,338	4,401	35	35	3	0.4	1,2,4,8,10	6,262	5,833	7.4	6,399	6,500	102	119	8	4.1
	5-50	1,2,10	4,322	4,345	-0.5	4,322	4,322	0	186	34	0.0	1,2,4,8	6,219	5,907	5.3	6,286	6,390	69	871	92	2.3
	10-10	1,2,10	4,322	4,122	4.9	4,366	4,542	98	12	2	1.0	1,2,4,7,8	6,449	5,742	12.3	6,602	6,870	160	31	1	7.4
	10-25	1,2,10	4,322	4,263	1.4	4,322	4,322	0	60	6	0.0	1,2,4,10	6,164	5,875	4.9	6,241	6,321	57	226	12	1.5
	10-50	1,2,10	4,322	4,336	-0.3	4,322	4,322	0	426	40	0.0	1,2,8,10	6,146	5,912	4.0	6,200	6,249	43	2,293	156	0.9
	20-10	1,2,10	4,322	4,105	5.3	4,322	4,322	0	26	3	0.0	1,2,4,8,10	6,262	5,889	6.3	6,427	6,576	133	44	5	4.6
	20-25	1,2,10	4,322	4,216	2.5	4,322	4,322	0	136	5	0.0	1,2,4,10	6,164	5,728	7.6	6,235	6,321	57	367	54	1.4
	20-50	1,2,10	4,322	4,297	0.6	4,322	4,322	0	635	85	0.0	1,2,8,10	6,146	5,784	6.3	6,154	6,164	10	9,911	1,131	0.1
SIBSAA	5-5	1,2,10	4,322	-	-	4,338	4,401	35	19	3	0.4	1,2,8,10	6,146	-	-	6,168	6,219	30	76	7	0.4
	5-10	1,2,10	4,322	-	-	4,322	4,322	0	49	6	0.0	1,2,8,10	6,146	-	-	6,150	6,164	8	165	20	0.1
	5-15	1,2,10	4,322	-	-	4,322	4,322	0	184	39	0.0	1,2,8,10	6,146	-	-	6,146	6,146	0	238	26	0.0
Exact	-	1,2,10	4,322	-	-	-	-	-	659	-	-	1,2,8,10	6,146	-	-	-	-	-	>21600	-	-

Table 2. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probabilities $q = 0.1$ and $q = 0.3$.

Table 3 results for $q = 0.5$ and $q = 0.7$ show that SAA is not able to find the optimum solution except for the sample $(M,N) = \{(20,50)\}$ and SIBSAA finds the optimum solution in all samples except for the sample $(M,N) = \{(5,5)\}$ when $q = 0.5$. With $q = 0.5$, the SIBSAA converged to the optimum solution with an optimality gap 0.4 in 119 seconds on the average. The SAA's average GAP_2 and CPU time are 4.4% and 1,499 seconds. With $q = 0.7$, the SIBSAA converged to the optimum solution with an optimality gap 0.4 in 69 seconds on the average. The SAA's average GAP_2 and CPU time are 2.4% and 132 seconds. These results show that SIBSAA is able to find better quality solutions in much less time than SAA.

		q=0.5										q=0.7									
		Best Solution				Randomized Trials						Best Solution				Randomized Trials					
Method	M-N	F^*_{Best}	Best Obj.	$\overline{v}^M$	GAP_1 (%)	Ave. Obj.	Worst Obj.	Stdev. Obj.	CPU(s) Ave	CPU(s) Stdev	GAP_2 (%)	F^*_{Best}	Best Obj.	$\overline{v}^M$	GAP_1 (%)	Ave. Obj.	Worst Obj.	Stdev. Obj.	CPU(s) Ave	CPU(s) Stdev	GAP_2 (%)
SAA	5-10	1,2,4,6,8	9,066	7,785	16.5	9,288	9,536	174	5	1	7.6	1,2,4,5,6,8,10	13,173	10,124	30.1	13,529	13,780	276	4	0	4.9
	5-25	2,4,7,8,10	9,003	7,953	13.2	9,147	9,276	133	106	14	5.9	1,2,4,6,7,8,10	13,117	10,895	20.4	13,457	13,883	334	27	4	4.3
	5-50	1,2,4,5,8,10	8,725	8,124	7.4	8,820	8,879	72	2,315	223	2.2	1,2,3,4,7,8,10	12,962	11,254	15.2	13,222	13,307	147	134	15	2.5
	10-10	1,2,3,4,7,8	8,991	7,541	19.2	9,234	9,357	151	11	1	6.9	1,2,3,4,8,10	13,019	10,283	26.6	13,284	13,613	252	6	1	3.0
	10-25	1,2,3,4,7,10	8,887	7,832	13.5	9,015	9,165	99	152	23	4.4	1,2,3,4,7,8,10	12,962	11,776	10.1	13,096	13,307	141	60	6	1.5
	10-50	1,2,7,8,10	8,746	8,448	3.5	8,811	8,874	57	4,088	354	2.0	1,2,4,5,7,8,10	12,939	12,326	5.0	13,049	13,307	148	263	25	1.1
	20-10	1,2,4,6,8,10	8,917	7,488	19.1	9,125	9,357	159	22	2	5.7	1,2,3,4,8,10	13,019	10,163	28.1	13,169	13,307	102	14	1	2.1
	20-25	1,2,4,7,8	8,811	7,875	11.9	8,979	9,088	101	314	53	4.0	1,2,3,4,7,8,10	12,962	11,670	11.1	13,069	13,173	97	132	14	1.3
	20-50	1,2,4,8,10	8,634	7,973	8.3	8,740	8,861	82	6,478	337	1.2	1,2,4,7,8,10	12,903	12,057	7.0	13,011	13,173	104	550	47	0.8
SIBSAA	5-5	1,2,4,7,10	8,710	-	-	8,725	8,746	20	51	6	1.1	1,2,4,7,8,10	12,903	-	-	13,011	13,173	104	26	3	0.8
	5-10	1,2,4,8,10	8,634	-	-	8,649	8,710	34	108	13	0.2	1,2,4,7,8,10	12,903	-	-	12,934	12,964	30	54	5	0.2
	5-15	1,2,4,8,10	8,634	-	-	8,644	8,685	23	197	14	0.1	1,2,4,7,8,10	12,903	-	-	12,917	12,939	20	126	15	0.1
Exact	-	1,2,4,8,10	8,634	-	-	-	-	-	>21600	-	-	1,2,4,7,8,10	12,903	-	-	-	-	-	>21600	-	-

Table 3. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probabilities $q = 0.5$ and $q = 0.7$.

Table 4 presents the results for failure probability $q = 0.9$. The SIBSAA converge to a solution with GAP_2=0.1% in 8 seconds, whereas the SAA's average GAP_2 and CPU time are 1.2% and 3 seconds.

q=0.9

		Best Solution				Randomized Trials					
Method	M-N	F^*_{Best}	Best Obj.	$\overline{v}^M$	GAP_1 (%)	Ave. Obj.	Worst Obj.	Stdev. Obj.	CPU(s) Ave	CPU(s) Stdev	GAP_2 (%)
SAA	5-10	2,7	20,701	17,254	20.0	20,830	20,922	100	1	0	1.9
	5-25	2,7	20,701	18,115	14.3	20,835	20,921	106	1	0	1.9
	5-50	2,4	20,589	19,216	7.1	20,631	20,701	57	2	0	0.9
	10-10	2	20,594	17,988	14.5	20,726	20,891	108	1	0	1.4
	10-25	1,2,4,8	20,567	18,322	12.3	20,604	20,701	55	2	0	0.8
	10-50	1,2,8,10	20,537	19,658	4.5	20,564	20,594	27	3	0	0.6
	20-10	1,2,7,10	20,657	18,223	13.4	20,747	20,892	110	2	0	1.5
	20-25	1,2,4,8	20,567	18,795	9.4	20,607	20,657	47	4	1	0.8
	20-50	1,2,8,10	20,537	19,778	3.8	20,578	20,657	46	12	1	0.7
SIBSAA	5-5	1,2,10	20,450	-	-	20,475	20,482	14	3	0	0.2
	5-10	1,2	20,439	-	-	20,451	20,479	17	9	2	0.1
	5-15	1,2	20,439	-	-	20,441	20,450	5	11	1	0.0
Exact	-	1,2	20,439	-	-	-	-	-	239		-

Table 4. Solution quality and CPU time performances of SAA and SIBSAA for CRFLP with facility failure probability $q = 0.9$.

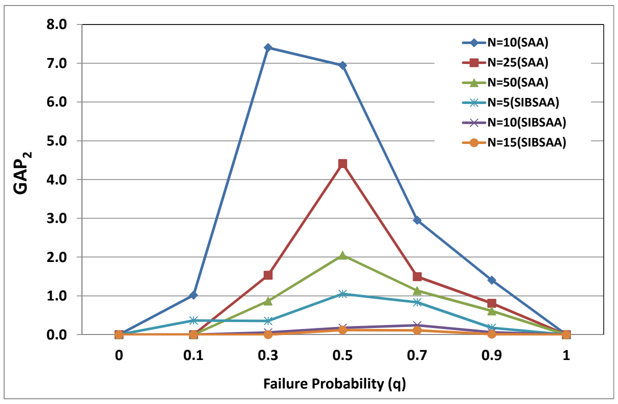
Figure 4a illustrates the effect of facility failure probability on the average solution quality (GAP_2) of SIBSAA with $N = 10$ in comparison with those of SAA with $N = 10$, 25 and 50 scenarios. The SIBSAA is always outperforming the SAA in terms of solution quality; the largest improvements are observed when the failure probability is neither too low nor too high.

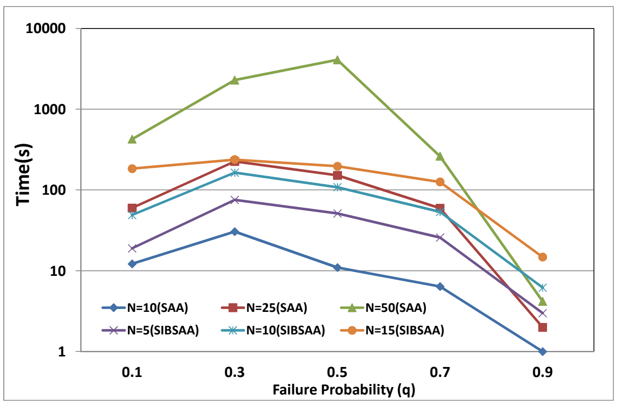
Figure 4b, a semi-log plot, illustrates the effect of facility failure probability on the CPU time performance of SIBSAA with $N = 5$, 10 and 15 in comparison with those of SAA with $N = 10$, 25 and 50 scenarios. We note the CPU time performance of SIBSAA with $N = 5$, 10 and 15 is comparable to that of SAA with $N = 25$ and significantly better than $N = 50$. Hence, in comparison with SAA, the SIBSAA improves the solution quality with similar computational effort. Moreover, the computational effort necessary to attain the same solution performance is much less with SIBSAA than SAA.

5. Conclusion

We developed a hybrid method, SIBSAA, which integrates the swarm intelligence concept of the PSO within the SAA methodology to efficiently solve the capacitated reliable facility location problems. This integration considers each sample solution as a particle in the swarm and employs the SAA algorithm iteratively. In each iteration, the social learning is injected into the solution process by introducing penalty terms in the objective that guides the solution of each sample to the swarm's balanced solution. The two key parameters of SIBSAA are swarm's learning bias and swarm's learning parameter. The learning bias parameter adjusts the importance given to swarm's best found solution and to the most recent average sample solution in calculating the swarm's balanced solution. The learning parameter modulates the rate at which the sample solutions converge to the swarm's balanced solution. Given that the swarm's best found solution improves over time, we propose two strategies for the bias parameter: static versus dynamic strategy.

We first conducted experiments for sensitivity analysis of the algorithm with respect to the parameters as well as number of samples. The results show that the SIBSAA's solution quality





(a) (b)

Figure 4. Solution quality (a) and CPU time (b) comparison of SIBSAA and SAA with different sample sizes (N) and failure probabilities (q). (M=10 for SAA and M=5 for SIBSAA)

performance is relatively insensitive to the choice of strategy for the swarm learning bias parameter (i.e., both the static and dynamic strategies are able to converge to the optimum solution). However, the dynamic strategy is slightly more computationally efficient than the static strategy. Further, increasing the number of samples improves the solution quality at a cost of linearly increasing computational effort. Lastly the SIBSAA is able to converge to the optimal solution even with a small number of samples. In addition to the sensitivity experiments, we compared the performance of SIBSAA with SAA's. These results show that the SIBSAA is able to improve the solution quality noticeably with reasonable computational effort compared to SAA. Further, increasing SAA's sample size to match the solution quality performance of SIBSAA requires significant computational effort which is not affordable for many practical instances of CRFLP.

There are three possible avenues of future research on SIBSAA. The first opportunity is to investigate the integration of alternative swarm intelligence mechanisms to improve the convergence rate and solution quality. Another extension is to investigate the conditions under which the dynamic strategy for swarm's learning bias is more advantageous than the static strategy. Lastly, the adaptation of the ideas of SIBSAA to multi-stage stochastic programs (e.g., reliable facility location over multiple periods) is another fruitful future research avenue.

Acknowledgments We are thankful to the editors and three reviewers for their feedback and comments, which helped improve the presentation clarity, quality of the methodological exposition and interpretation of the computational results.

REFERENCES:

Ahmed, S., Shapiro, A., 2002. The sample average approximation method for stochastic programs with integer recourse. Optimization Online, http://www.optimization-online.org/.

Arostegui, Jr. M.- A., Kadipasaoglu, S.-N., Khumawala, B.-M., 2006. An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of Production Economics, 103 (2), 742-754.

Brennan, P., 2011. Lessons learned from the Japan earthquake. Disaster Recovery Journal, 24(3), http://www.drj.com/2011-articles/summer-2011-volume-24-issue-3/lessons-learned-from-the-japan-earthquake.html. Last accessed on Sunday, February 05, 2012.

Chouinard, M., D'Amours, S., Aït-Kadi, D., 2008. A stochastic programming approach for designing supply loops. International Journal of Production Economics. 113, 657-677.

Gade, D., 2007. Capacitated facilities location problems with unreliable facilities. Master's Thesis, University of Arkansas.

Higle, J.-L., Sen, S., 1991. Stochastic decomposition: An algorithm for two-stage linear programs with re-course. Mathematics of Operations Research, 16, 650- 669.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. The Proceedings of the International Conference on Systems, Manand Cybernetics, 5, IEEE Press, 4104-4108.

Kennedy, K., Eberhart, R., 1995. Particle swarm optimization. In The Proceedings of the 1995 IEEE International Conference on Neural Network, 1942-1948.

Kleywegt, A.-J., Shapiro, A., Homem-De-Mello, T., 2001. The sample average approximation method for stochastic discrete optimization. SIAM Journal of Optimization, 12, 479-502.

Kratica, J., Tosic, D., Filipovic, V., Ljubic, I., 2001. Solving the simple plant location problems by genetic algorithm. RAIRO Operations Research, 35, 127-142.

Linderoth, J., Shapiro, A., Wright, S., 2006. The empirical behavior of sampling methods for stochastic programming. Annals of Operations Research, 142(1), 215-241.

Masihtehrani, B., 2011. Stochastic analysis of disruption in supply chain networks. PhD Dissertation, Pennsylvania State University, USA.

Peng, P., Snyder, L.W., Lim, A., Liu, Z., 2011, Reliable logistics networks design with facility disruptions, Transportation Research Part B, 45, 1190-1211

Rockafeller, R.-T., Wets, R.-J.-B., 1991. Scenarios and policy aggregation in optimization under uncertainty. Mathematics and Operations Research, 16 (1), 119-147.

Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A., 2005. A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research, 167, 96-115.

Schütz, P., Tomasgard, A., Ahmed, S., 2009. Supply chain design under uncertainty using sample average approximation and dual decomposition. European Journal of Operational Research 199 (2), 409-419.

Shapiro, A., Homem-de-Mello, T., 1998. A simulation-based approach to two-stage stochastic programming with recourse. Mathematical Programming, 81, 301-325.

Shen, Z.-J.-M., Zhan, R.-L., Zhang, J., 2011. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. Informs Journal on Computing, 23(3), 470-482.

Shi, Y., Eberhart, R.-C., 1998. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation, 69-73.

Snyder, L.-V., Daskin, M.-S., 2005. Reliability models for facility location: The expected failure cost case. Transportation Science, 39, 400-416.

Snyder, L.-V., Scaparra, M.-P., Daskin, M.-S., Church, R.-L., 2006. Planning for disruptions in supply chain networks. Forthcoming in Tutorials in Operations Research, Informs, Baltimore, MD, USA.

Snyder L.-V., Ülker, N., 2005. A model for locating capacitated, unreliable facilities. IERC Conference, Atlanta, GA, USA.

The Guardian, 2011. Toyota profit slides on Japan earthquake disruption. http://www.guardian.co.uk/business/2011/may/11/toyota-profit-hit-by-japan-earthquake. Last accessed on Sunday, February 05, 2012.

Van Slyke, R., Wets, R.-J.-B., 1969. L-shaped linear programs with applications to optimal control and stochastic linear programs. SIAM Journal on Applied Mathematics, 17, 638-663.

Verweij, B., Ahmed, S., Kleywegt, A., Nemhauser, G., Shapiro, A., 2002. The sample average approximation method applied to stochastic routing problems: A computational study. Technical report, Georgia Institute of Technology, Atlanta, GA, USA.

Wang, K.-J., Wang, S.-M., Chen, J.-C., 2008. A resource portfolio planning model using sampling-based stochastic programming and genetic algorithm. European Journal of Operational Research, 184, 327-340.

Zhan, R.-L., 2007. Models and Algorithms for Reliable Facility Location Problems and System Reliability Optimization, PhD Dissertation, University of Florida.