CHAPTER 1
INTRODUCTION

1.1 Importance of supply chain management (SCM)

Long-term competitiveness and increase in global market share of a company depend on to what extent it meets customer preferences in terms of service, cost, delivery lead time, quality, flexibility and so on, which in turn depends on the fact that whether its supply chain (SC) is designed more effective and efficient than its competitors’. SC is said to be efficient if it can attain required outcomes with minimum use of resources and effective if it delivers products or services in a manner that is acceptable to end-users (Stern et al., 1996). Efficiency is measured by total cost of production, inventory level, delivery performance, and effectiveness is measured by service quality and service needs of customers (Mentzer, 1999).

1.2 Logistics v/s SCM

Literature of nineteenth century establishes that Logistics is concerned with the organization of moving, storing and supplying troops and necessary equipment and hence finds application in military and mathematical domains. But today it spans much beyond that. In 1970s, oil crises resulted in, hike in both transportation costs and interest rates of inventory carrying. This made the top management to understand the role of logistics in increasing the company’s profits. As a result, integration approach is adopted for physical distribution and purchasing with material handling and its optimization. This in turn led to the evolution of logistics management which encompasses physical distribution and materials management (Ballou, 1992). The rapid development in information technology, was acting as catalyst to speed up this
integration process. It was in late 1980s that the term SCM found its place in the literature. This term got popularity both in business and literature domains when Cooper et al., (1997) addressed it as the extension of logistics. The Council of Logistics Management, an international organization of logistics professionals, has defined logistics as: “The process of planning, implementing and controlling the efficient, cost-effective flow and storage of raw materials, in-process inventory, finished goods and related information from point of origin to point of consumption for the purpose of conforming to customer requirements.”

According to Christopher (1998) logistics management is, “The process of strategically managing the procurement, movement and storage of materials, parts and finished inventory (and the related information flows) through the organization and its marketing channels in such a way that current and future profitability are maximized through cost-efficient fulfillment of orders.” Focus on end-customer value and the coordination of activities and processes within and between organizations in the SC that makes it to extend beyond logistics (Bowersox et al., 2000).

1.3 The scope and functions of SCM

The conceptual model of SCM developed by (Mentzer et al., 2001) is shown in Fig. 1.1. A SC can be illustrated as a pipeline, with directional SC flows of products, services, financial resources, and the informational flows of demand and forecasts. The traditional business functions of marketing, sales, research and development, forecasting, production, procurement, logistics, information technology, finance, and customer service manage and accomplish these flows from the supplier’s suppliers through the customer’s customers to ultimately provide value and enhance customer satisfaction. The Fig. also shows the critical role of customer value and satisfaction to
achieve competitive advantage and profitability for the individual companies in the SC, and the SC as a whole.

1.4 SC planning framework

The SC is a complex network of facilities and organizations, with different and conflicting objectives. Hence to facilitate effective decision-making, planning processes are typically subdivided into multiple hierarchical-based planning levels. Each level has a planning cycle that its processes follow. Currently, SC planning decisions are usually made using three hierarchical planning levels:

- Strategic or high-level planning - done yearly or on an ad hoc basis
- Tactical or mid-level planning - done quarterly or monthly
- Operational or low-level planning - weekly, daily, or by shift

![Diagram of SC Planning Framework]

Figure 1.1 A model of SCM (Mentzer et al., 2001)

1.4.1 Strategic Level Planning

In the SC design phase, strategic decisions, such as facility location decisions and technology selection decisions play major roles. Optimization tools are used to
determine the location, size, and the number of plants, distribution centers, and suppliers. Generally, supply chain network (SCN) design is done once in few years as companies do not need to add new plants or distribution centers on a routine basis.

1.4.2 Tactical Level Planning

SC planning at a tactical level involves optimizing the flow of goods throughout a given SC configuration over a time horizon. Having decided SC design, supply planning develops sourcing, production, deployment, and distribution plans. Inventory management decisions on raw materials, intermediate products, and end products and distribution decisions within the SC are fewer decisions to name that are done in tactical level planning.

1.4.3 Operational Level Planning

At an operational level, scheduling is done for all resource needs, including labor, equipment, and materials. Generally, production scheduling is done frequently, potentially several times a day to account for changes to orders, machine failures, material shortages, and other plant disruptions. (Chopra and Meindl, 2005).

1.5 SCN optimization

1.5.1 About optimization

Over the second half of the 20th century, optimization gained popularity, in different engineering disciplines such as in the study of physical and chemical systems, production planning and scheduling systems, location and transportation problems, resource allocation in financial systems, and engineering design (Sahinidis, 2004). Optimization is the act of obtaining the best results for the given problem under given circumstances. An optimization problem can be stated as follows:
Find $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ which minimizes $f(X)$ \hspace{1cm} \ldots (1.1)

Subject to constraints

$g_j(X) \leq 0, \quad j = 1, 2, \ldots, m$

$l_j(X) = 0, \quad j = 1, 2, \ldots, p$

In equation (1.1), ‘$X$’ is a ‘$n$’ dimensional vector called the design vector, $f(X)$ is termed as the objective function, and $g_j(X)$ and $l_j(X)$ are known as inequality and equality constraints respectively. Optimization problems can be broadly classified into continuous and discrete problems. The design variables $x_1, x_2, \ldots, x_n$ of an optimization problem are permitted to take any real value and then the optimization problem is called as continuous optimization problem. Continuous problems are constrained and unconstrained in nature. Most of the real world unconstrained and constrained problems are highly nonlinear, multimodal and multidimensional in nature. An optimization problem with discrete variables is known as a combinatorial optimization problem. In a combinatorial optimization problem one is looking for an integer, from a finite set. Linear integer programming refers to the class of combinatorial constrained optimization problems with integer variables, where the objective function is a linear function and the constraints are linear. A wide variety of real life problems in logistics, such as the warehouse location problem, travelling salesman problem, decreasing costs and machinery selection problem, network and graph problems, such as maximum flow problems, set covering problems, spanning tree problems and many scheduling problems etc. can be formulated as linear integer optimization problems.
1.5.2 Optimization solution techniques

Compared to Linear Programming where, due to the convexity of the problem, any local solution is a global optimum, the integer programming problems have many local optima and finding a global optimum to the problem requires one to prove that a particular solution dominates all the feasible points by arguments other than the calculus based derivative approaches of convex programming with continuous variables (Krasimira Genova and Vassil Guliashki, 2011). Most Combinatorial Optimization Problems (COPs) are difficult to solve and are NP-hard (Non-deterministic Polynomial) (Garey and Johnson, 1979). Owing to the complexity and the practical importance of NP-hard COPs, a number of techniques have been proposed for solving such problems in literature. All these techniques for solving COPs can roughly be classified into two main categories: exact and heuristic algorithms. Exact methods can be used only for small instances and simple problems. Because they cannot solve complicated problems in time, depending polynomially on the problem input data length. These algorithms if used require a number of computational steps that grow exponentially with size of problem. The exponential growth of computational effort required means that, optimal solution cannot be found within a reasonable amount of computational time (Foulds, 1983). A diagrammatic representation of classification of optimization methods is shown in Fig. 1.2.

1.5.3 Exact Optimization Methods

Exact algorithms are guaranteed to find an optimal solution for every instance of a COP with the disadvantage that the run-time, increasing dramatically with a problem instance’s size. Only small or moderately-sized instances can be practically solved to optimality. Krasimira Genova and Vassil Guliashki, (2011) summarize that there are,
at least, three different approaches for solving integer programming problems using
exact methods

1. Cutting plane algorithms
2. Enumerative approaches and Branch-and-Bound, Branch-and-Cut and
   Branch-and-Price methods
3. Relaxation and decomposition techniques.

Figure 1.2 Classification of Optimization methods

1.5.4 Approximate Methods of optimization

For larger instances of COPs, the only possibility is the application of
approximate algorithms. Though the solution obtained using these methods is sub-
optimal when compared to exact methods, give satisfactory performance and
manageable when applied to real-life applications. They are not so sensitive to little
changes in some constraints (Krasimira Genova and Vassil Guliashki, 2011). These
methods focus on escaping local optima and try to find the global optimum solution.
The major advantage of these algorithms is that their efficiency or applicability is not
tied to any specific problem domain. These methods are more popularly known as heuristics.

The term heuristic means a method which, is seeking good (i.e. near-optimal) solutions at a reasonable computational cost on the basis of experience or judgment, and yields a reasonable solution to a problem, but cannot be guaranteed to produce the mathematically optimal solution (Reeves, 1995). Foulds (1983) summarizes following reasons for utilizing heuristic solution methods.

1. Decision rules of decision makers are facilitated by heuristic rules than with a complex optimization routine.
2. Managers may be quite satisfied with a heuristic solution that produces better results than those currently achieved.
3. Fast, reasonable, results are needed than optimization routines for real life applications.
4. Heuristics can be less sensitive to variations in problem characteristics and data quality.

Many researchers have focused their attention on a new class of algorithms called meta-heuristics. According to Osman (2002), “A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to produce efficient high-quality solutions by combining intelligently different concepts to explore and exploit the search space using adaptive learning strategies and structured information”. Meta-heuristics are designed with not getting trapped in a local optimum and reducing the search space. Each meta-heuristic has one or more adjustable parameters to incorporate flexibility. The use of meta-heuristics has significantly produced good quality solutions to hard combinatorial problems in a reasonable time. The most familiar and powerful meta-heuristics are Simulated Annealing and Tabu Search, Guided Local Search, Iterated Local Search and Variable
Neighborhood Search. The Population-based algorithms are a large group of meta-heuristics based on the natural practices of surviving of the best that have a learning capability. These include: Genetic Algorithms, Scatter Search, Ant Colony Optimization, Particle Swarm Optimization and Memetic Algorithms (Krasimira Genova and Vassil Guliashki, 2011). The most important meta-heuristic approaches and algorithms are briefly considered below.

1.5.5 Simulated Annealing

Simulated annealing is, one of the oldest meta-heuristics designed to permit escaping from local optima. It is proposed by Kirkpatrick, et al., (1983) in 1983 and independently applied to travelling salesman problem by Cerny in 1985. The Simulated Annealing extends the local search allowing movement towards worse solutions. The basic algorithm of Simulated Annealing is presented by Dowsland (1993). The probability of accepting the movement to $x_{k+1}$ decreases with the increase of the deterioration $\Delta = f(x_{k+1}) - f(x_k)$ of the objective function or with the decrease of the temperature $T_k$. The control of the possibility for accepting the new solution is realized by means of the parameter $T_k$, the idea for which arises from the physical annealing process. The probability of movement towards a worse solution is high, and then it is gradually decreased during the subsequent search process. (Krasimira Genova and Vassil Guliashki, 2011).

1.5.6 Tabu search

The main idea of this algorithm was first introduced by Glover (Glover, 1986). The basic algorithm includes a local search with the greatest improvement and a short term memory to avoid the local optima and long term memory. The short term memory is applied as a Tabu list, where the last solutions considered are stored and
the movements directed towards them are forbidden. The neighborhood of a current solution includes only solutions, which are not in the Tabu list. The set of these solutions is called allowed set. At each iteration the best solution of this set is chosen as a new current solution. This solution is included in the Tabu list and one of the solutions stored in it is removed. In order to avoid the local optimum, the movement towards worse neighbor solutions is allowed. The search procedure is terminated after executing a given common limit of iterations or after a given number of consecutive iterations without improving the best obtained solution. (Krasimira Genova and Vassil Guliashki, 2011).

1.5.7 Population-based algorithms

These are meta-heuristics that handle a population of solutions in each iteration. These EAs mimic the natural evolutionary processes. At each iteration a set of operators is applied on the individuals in the current population in order to generate the individuals of the next generation. The fitness value of each solution in the population is evaluated. The individuals, having the highest fitness are used in the next population direct or as parents generating new individuals through a change or by means of a combination between them. The operators used are: modification or mutation which changes individuals directly, and combination or crossover between two or more individuals for generating new individuals. The remaining solutions (individuals) are rejected based on selection process. The EAs are nondeterministic algorithms. They differ from one another in way of their presentation, evaluation, selection and change of solutions. EAs include genetic algorithms, evolutionary programming, evolutionary strategies, genetic programming etc.

Initial pioneering works on Genetic algorithms (GA) can be listed as Fogel et al. (1966), Holland, (1962), Rechenberg, (1965) and have been published in the mid
sixties, but they have been further developed by Holland (1975) and Goldberg (1989). In Genetic algorithms (GA) genetic stands for ‘evolution’ as their mechanism mimics the genetic evolution of species. GA uses a population of feasible solutions, called individuals. A number of pairs of individuals are selected from the current population by means of a selection operator. Each pair performs reproduction by means of a crossover operator and generates two new individuals (solutions), called an offspring. A mutation operator is used to modify randomly with a small probability the offspring individuals imitating the mutation during the natural evolution. At the end the population individuals, having worse objective function values are replaced by the corresponding better offspring individuals. This procedure is iteratively repeated and it usually stops when the population does not improve anymore or after a fixed number of iterations.

The so called Ant systems have been proposed by Colomi, Dorigo and Maniezzo (1991) in 1991. It is inspired from the behavior of ants, searching food in the neighborhood of their formicary using the best route to the food source. Each ant is a constructive procedure that is able to generate a new solution on the base of two factors: the trace factor and the desirability factor. The first factor reflects the historical information gathered throughout the individual search of the ants and the second factor guides each ant to the choice of a solution with the best objective function value in its neighborhood. After each iteration the ants share their new information to update the trace factor. Since this work uses Particle Swarm Optimization (PSO) as optimizer it is dealt in detail in next section.

1.6 PSO Algorithm

PSO is first developed by Eberhart and Kennedy (1995), in 1995 and other references include Kennedy, and Eberhart (1995), Kennedy, and Eberhart (1997),
and Kennedy et al., (2001). The basic idea in PSO is to imitate the intelligent swarming behavior, observed in flocks of birds, schools of fish, swarms of bees, etc. PSO models a set of potential problem solutions as a swarm of particles moving about in a virtual search space. Each object (particle) in PSO moves from its current position to a new position based on sum of three vectors: inertia, competition and cooperation. The inertia vector is determined by the current velocity of the particle weighted by a constant w. In this way the tendency of the particle to maintain its current velocity is reflected. The competition vector links the current position of the particle to its personal best position found during the search process. This vector is weighted using a uniformly distributed random function. The cooperation vector links the current position of the particle to the global best position, found by the particles. This vector is weighted using a second uniformly distributed random function. Thus the cooperation among particles is important for finding the global optimum solution and the inertia and the competition are necessary for the particle to avoid trapping in the local minima.

The PSO search algorithm is initialized with a population of random solutions called particles. Each individual particle has no mass and volume but associated with a velocity. The trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of the other particles in the search space. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

According to Angeline (1998), the two main distinctions between PSO and EAs are:

1) EAs rely on three mechanisms while searching: parent representation, selection of individuals and the fine tuning of their parameters. In contrast, PSO relies on
only two mechanisms. The absence of selection mechanism in PSO is compensated by the use of leaders to guide the search. Compared to EAs, offspring generation is also not there in PSO.

ii) A second difference between EAs and PSO is regarding the way in which the individuals are manipulated. PSO uses the velocity of a particle as a directional mutation operator in which the direction is defined by both the particle’s personal best and the global best of the swarm. In contrast, EAs use a mutation operator that can set an individual in any direction.

Following are the two key aspects by which PSO has become more popular:

- The algorithm of PSO is relatively simple since in its original version, it only adopts one operator for creating new solutions, (unlike most EAs) and its implementation is straightforward and easy.

- PSO has been found to be very effective in a wide variety of applications, being able to produce very good results at a very low computational cost.

- Quick convergence can be achieved by adjusting easily few parameters in PSO algorithm to have more global search ability at the beginning of the run called exploration and have more local search ability near the end of the run called exploitation (Kennedy et al., 2001).

1.6.1 Terminology used in PSO algorithm

Terms used in PSO algorithm are listed here:

Particle: Member (individual) of the swarm. Each particle represents a potential solution to the problem being solved.

Swarm: Population of individual particles of the algorithm.
pbest (personal best): It is the best position of a given particle, so far it has attained. That is, the position of the particle that has provided the greatest success. $P_{kd}$ in equation 1.2 represents Pbest of $k^{th}$ particle in $d^{th}$ dimension.

Gbest (Global best): Position of the best particle of the entire swarm. $G_d$ in equation 1.2 represents Global best in $d^{th}$ dimension.

Leader: Particle that is used to guide another particle towards better regions of the search space.

Velocity (vector): This vector drives the optimization process, that is, it determines the direction in which a particle needs to “fly” (move), in order to improve its current position.

Inertia weight($w$): It is employed to control the impact of the previous history of velocities on the current velocity of a given particle and denoted by $w$.

Learning factor: It represents the attraction that a particle has towards either its own success or that of its neighbors. There are two learning factors used: $c_1$ and $c_2$, where $c_1$ is the cognitive learning factor and it represents the attraction that a particle has towards its own success and $c_2$ is the social learning factor and represents the attraction that a particle has toward the success of its neighbors. Both, $c_1$ and $c_2$, are constants (Reyes-Sierra and Coello Coello, 2006).

The new velocity and new position of the particle is calculated by using the following equations,

$$\text{Velocity } \mathbf{v}_{kd}^{\text{new}} = w \times \mathbf{v}_{kd} + \left[ c_1 \times \left( r_1 \times \left( P_{kd} - X_{kd} \right) \right) + c_2 \times \left( r_2 \times \left( G_d - X_{kd} \right) \right) \right]$$

for $d = 1,2,..,D.$  \hspace{1cm} (1.2)

$$\mathbf{X}_{kd}^{\text{new}} = \mathbf{X}_{kd} + \mathbf{v}_{kd}^{\text{new}}$$

\hspace{1cm} (1.3)

The equations (1.2) and (1.3) describe the flying trajectory of a population of particles. Equation (1.2) describes how the current velocity $v_{kd}$ is dynamically updated
and equation (1.3) describes the position update of current position $X_{k,d}$ the flying particle. The equation (1.2) consists of three parts, namely momentum part, cognitive part and social part. The balance among these three parts determines the global and local search ability, and hence the performance of PSO. The velocity and position updates based on swarm and personal influence was first depicted pictorially by Hassan et al., (2005) and is shown in Fig. 1.3. Later a new parameter called inertia weight ‘w’ is added into the original PSO algorithm. A large inertia weight facilitates a global search, while a small inertia weight facilitates local search (Shi and Eberhart, 1998). In equation (1.3), particles’ velocities on each dimension is clamped to a maximum velocity $v_{\text{max}}$, so that if the sum of the three parts on the right side exceeds a constant value specified by the user, then the velocity on that dimension is assigned to be $\pm v_{\text{max}}$. Here $r_1$ and $r_2$ represent the uniform random numbers.

PSO can be applied, like other algorithms in the field of evolutionary computation, in the areas of solving discrete problems involving system design, multi-objective optimization, classification, pattern recognition, system modelling, scheduling, planning, robotic applications, decision making, simulation and identification (Shi, 2004).

![Diagram](image)

Figure 1.3 Depiction of the velocity and position updates in Particle Swarm Optimization (Hassan et al., 2005)
1.7 Optimization procedure

Optimization is a stepwise procedure which starts with development of models. Models are the data-independent mathematical equations that describe the relationships among decisions, constraints, and objectives. The model must represent the real world to the degree needed to achieve accuracy. Once an optimization problem is formulated, an algorithm/solver determine the best course of action. A solver comprises a set of logical steps in a computer program to search for a solution that achieves the objective. A solver can develop two types of solutions: a solution that satisfies all constraints and called a feasible solution and feasible solution that achieves the best objective functional value (as per minimizing or maximizing) called an optimal solution.

1.8 SCN optimization and Operations Research

SC optimization tools enable companies to explore, and take advantages of, opportunities for improving the efficiency and effectiveness of their SCNs activities.

An optimization problem on SCN architecture comprises of mainly following decision variables and constraints. Decision variables could be:

- When and how much of a raw material to order from a supplier?
- When to manufacture an order?
- When and how much of product to ship to customer or distribution center?

SCN Constraints or limitations could be:

- A supplier's capacity to produce raw materials or components
- A specified number of hours per day for which a production line can run
- A distribution center's storage capacity.
Constraints in an optimization problem could be either hard or soft. Examples for hard constraints are the number of working hours in a shift or the maximum capacity of a truck, etc. These are stringent and cannot be altered. Soft constraints can be relaxed or violated. Examples of soft constraints include customer due dates or warehouse space limitations. Customer due dates can be changed or a product may be squeezed into a warehouse temporarily, making constraints less stringent.

General Optimization Objectives of SCN are to maximize, minimize, or satisfy something, such as the following:

- Maximizing profits
- Minimizing SC costs
- Maximizing customer service
- Minimizing lead time
- Maximizing production throughput

The use of meta-heuristics has significantly produced good quality solutions to hard combinatorial problems in the context of SCN optimization in a reasonable time.

1.9 The multi-objective optimization problem

1.9.1 Basic concepts of multi-objective programming

The general form of multi-objective optimization is as follows:

Minimize/Maximize : \( f_m(x), \quad m = 1, 2, \ldots, M; \)

Subject to : \( g_j(x) \geq 0, \quad j = 1, 2, \ldots, J; \)

\( h_k(x) = 0 \quad k = 1, 2, \ldots, K; \)

\( x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, 2, \ldots, n; \)

Here \( m \) represents number of objectives, \( j \) number of inequality constraints, \( k \) number of equality constraints. The last set of constraints is called variable bounds, restricting each decision variable \( x_i \) to take a value within a lower bound \( x_i^{(L)} \) and an upper
bound $x_i^{(U)}$. According to Deb, (2001), there are three fundamental differences between single-objective and multi-objective optimization problems.

- Multi-objective optimization has two or more goals, whereas single-objective problem has only one goal. Progressing towards the Pareto-optimal front and maintaining a diverse set of solutions in the non-dominated front are important two goals.

- Multi-objective optimization involves two search spaces, instead of one. They are decision variable space, and the objective space. These two spaces are related by a unique mapping between them, and the properties of the two search spaces are not similar. For example, proximity of two solutions in one space does not mean proximity in the other space.

- Multi-objective optimization for finding multiple Pareto-optimal solutions eliminates all artificial fix-ups, such as in the weighted sum approach and $\varepsilon$-constraint method. It finds a set of optimal solutions corresponding to different weight and $\varepsilon$-vector in one single simulation run.

1.9.2 Multi-objective optimization solution methodology

Since multi-objective optimization formulation has more than one objective to be optimized simultaneously, there cannot be a single optimal solution which simultaneously optimizes all objectives. The resulting outcome is a set of optimal solutions with a varying degree of objective values. This set of solutions is called non-dominated set. The following two conditions must be true for a non-dominated set:

1. Any two solutions of non-dominated set must be non-dominated with respect to each other.

2. Any solution not belonging to non-dominated set is dominated by at least one member of non-dominated set.
Such a non-dominated set is also called as Pareto-Optimal Set. Thus multi-objective optimization problem has many optimal solutions. Many methodologies have been proposed in literature, for treating multi-objective optimization problems. Among them, the weighted-sum method, the $\varepsilon$-constraint method, and the goal-programming method etc. convert vector objectives into a scalar objective. The resulting problem can then be solved by using any existing optimization technique. If the algorithm does not converge to a suitable solution or the decision maker (DM) does not agree with the result, the DM need to adjust the related parameters, such as weighting factors in the weighted-sum method. The computation needs to be repeated until a satisfactory solution is obtained. This is an interactive process, and the DM needs to be constantly involved in the decision (Chen, 2003).

From the above discussion it is clear that the optimization of a multi-objective problem is a procedure looking for a compromise policy, consists of an infinite number of options. Analyzing the system using traditional optimization techniques may mislead to sub-optimal results as all these methods are subjective and convert a multi-objective optimization problem into a single-objective optimization problem. Since then there has been increasing interest to use EAs in solving multi-objective optimization because of their ability to find multiple optimal solutions in single run and handle efficiently complexity of NP-hard problems. The ideal situation is that the decision maker should be presented with a vector of optimal solutions. The final decision is made among them taking the total balance overall criteria into account. This balancing over criteria is called trade-off and this trade-off level may change over time due to uncertainty and global competitiveness. Hence the SC performance needs to be evaluated continuously and SC managers should make timely and right decisions (Shen, 2007).
1.9.3 Constraint Handling

A constraint is a condition that a solution to an optimization problem is required to satisfy. Constraints divide the search space into two divisions-feasible and infeasible regions. The set of candidate solutions that satisfy all constraints is called the feasible set. If a constraint is not satisfied at a given point, the point is said to be infeasible. Most of the real-world optimization problems are constrained problems. Different methods of handling constraints are listed below.

- Preserving feasibility of solutions
- Penalty function approach
- Decoder-based methods
- Hybrid methods (with classical approaches)
- Separation of objectives and constraints
- Repair algorithms

1.9.4 Penalty functions Method

This is the most popular approach used in EA literature to handle constraints. In this method, the constrained problem is transformed to an unconstrained one, by penalizing the constraints and building a single objective function, which in turn is minimized using an unconstrained optimization algorithm. Penalizing constraints are done by adding or subtracting to/from the objective function based on amount of constraint violation. In optimization, two types of penalty functions are considered. They are exterior and interior. In case of exterior methods search starts with an infeasible solution and from there movement is towards feasible region. In case of interior points, penalty term is tuned such that its value will be small at points away from the constraint boundaries and tend to infinity as constraint boundaries are approached. Thus in this method, search process starts with a point in a feasible
region, and stays within the feasible region as the constraint boundaries act as barriers. The exterior penalty method does not require an initial feasible solution. In contrast, the interior penalty method needs initial feasible solution finding which itself could be NP-hard in nature. Hence the exterior penalty method is more popular in EAs in literature (Deb, 2000).

Exterior penalty methods require setting penalty parameters for individual constraint functions. These selected values affect the convergence performance. Thus selecting suitable values in cases of many practical applications is too difficult. In particular, under-penalization of the infeasible solutions (i.e; applying too low weights for each constraint function) results in too slow convergence towards the feasible solutions, or no feasible solution. Over-penalization (i.e; applying too high weights for each constraint function) results in a rapid convergence to a feasible solution. These may lead to premature convergence to a suboptimal solution. If the number of constraint functions is relatively high, i.e; >10, finding the acceptable penalty parameter values is a difficult task as the user does not have enough problem specific information available for selecting these values in advance. So, the only alternative is to set the values for the penalty parameters by applying an educated guess and then refine the values by the trial-and-error method (Lampinen, 2001).

The general penalty function method of constraint handling runs like this: Before the constraint violation is calculated, all constraints are normalized. Thus, the resulting constraint functions are \( g_j(x^{(0)}) \geq 0 \) for \( j = 1,2,\ldots,\ldots, J \). For each solution \( x^{(i)} \), the constraint violation for each constraint is calculated using equation (1.4).

\[
\omega_j (x^{(i)}) = \begin{cases} 
|g_j(x^{(i)})| & \text{if } g_j(x^{(i)}) < 0 \\
0 & \text{otherwise}
\end{cases} \quad \ldots(1.4)
\]

Thereafter, all constraint violations are added together to get the overall constraint violation using equation 1.5.
\[ \Omega(x^{(0)}) = \sum_{i=0}^{n} \omega_i(x^{(0)}) \]

\[ \text{(1.5)} \]

This constraint violation is then multiplied with a penalty parameter \( R_m \) and the product is added to each of the objective function values to get modified objective function given in equation 1.6.

\[ F_m(x^{(0)}) = f_m(x^{(0)}) + R_m \Omega(x^{(0)}) \]

\[ \text{(1.6)} \]

The function \( F_m \) takes into account the constraint violations. For a feasible solution, the corresponding \( \Omega \) term is zero and \( F_m \) becomes equal to the original objective function \( f_m \). However, for an infeasible solution, \( F_m > f_m \). The penalty parameter \( R_m \) is used to make both of the terms on the right side of the above equation to have the same order of magnitude.

Pros and Cons of Penalty Functions can be summarized as follows:

- The main advantage of this method is its generality and simplicity. This method is known as the most common approach for handling linear and nonlinear constraints in Evolutionary optimization.
- The penalty function approach is readily available method for converting constrained problems of any type into unconstrained problems.
- Many constraints in the real world are "soft", in the sense that they need not be satisfied precisely. The penalty function approach is best-suited for this type of problems.
- The drawback with penalty function methods is that the solution to the unconstrained penalized problem may not be an exact solution to the original problem.
- In some cases penalty methods can't be applied because the objective function is undefined outside the feasible set.
Another drawback to penalty methods is that as the penalty parameters are increased to more strictly enforce the constraints, the unconstrained formulation becomes complicated, with large gradients and abrupt function changes.

1.10 Location-Allocation Decisions in SCM

In the SC design phase, strategic decisions, such as facility location decisions play a major role. It is very important to design an efficient SC to facilitate the movements of goods. These strategic decisions lead to costly, time consuming investment as the facilities located today, are expected to remain in operation for a long time. Environmental changes during the facility's lifetime can drastically alter the appeal of a particular site, turning today's optimal location into tomorrow's investment blunder. Also SC configuration decides whether it is a responsive or efficient SC. Determining the best locations for new facilities is thus an important strategic challenge (Owen and Daskin, 1998). During SCN design, in one of the phases, managers must decide on location and capacity allocation for each facility which can be made jointly. The demand allocation decisions can be altered on a regular basis as costs change and markets evolve (Chopra and Meindl, 2005).

1.11 Location-allocation decisions using Hybrid PSO

1.11.1 Particle Swarm Model for Binary Decision

Kennedy and Eberhart (1997) proposed a discrete binary version of PSO for binary problems. Their model proposes that the probability of an individual’s deciding yes or no, true or false or making some other binary decision, is a function of personal and social factors and is given by equation (1.7).
\[ P(x_{id}(t) = 1) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd}) \] \hspace{1cm} \text{(1.7)}

Where

- \( P(x_{id}(t) = 1) \) is the probability that individual \( i \) will choose 1.
- \( x_{id}(t) \) is the current state of individual \( i \).
- \( t \) is the current time step, and \( t-1 \) is the previous step.
- \( v_{id}(t-1) \) is a measure of the individual’s current probability of deciding 1.
- \( p_{id} \) is the best state found so far, for example, it is 1 if the individual’s best success occurred when \( x_{id} \) was 1 and 0 if it was 0. It is referred as Pbest.
- \( p_{gd} \) is the neighborhood best, again 1 if the best success attained by any member of the neighborhood was when it was in 1 state and 0 otherwise. It is referred as gbest.

The parameter \( v_{id}(t) \), an individual’s inclination to make one or the other choice, will determine the probability threshold. If \( v_{id}(t) \) is higher, the individual is more likely to choose 1, and lower values favor the 0 choice. Such a threshold stays in the range [0.0, 1.0]. The sigmoid function is a logical choice to do this. It squashes the range of \( v_{id} \) to a range of [0.0,1.0]. The sigmoid function used to determine the probability threshold is given by equation (1.8),

\[ g(v_{id}) = \frac{1}{1+\exp(-v_{id})} \] \hspace{1cm} \text{(1.8)}

In any situation, whether individual learning (Pbest) is stronger or social-influence (Gbest) is stronger is unknown, both will be weighed by random numbers, so that sometimes the effect of one and sometimes the other will be stronger. The symbol \( \phi \) is used to represent a positive random number drawn from a uniform distribution with a predefined upper limit so that the two \( \phi \) limits sum to 4. Thus formulae for binary decision are given by equations (1.9) and (1.10).
If \( r_{id} < s(v_{id}(t)) \) then \( x_{id}(t) = 1 \); else \( x_{id}(t) = 0 \)...

Then \( r_{id} \) is a vector of random numbers, drawn from a uniform distribution between 0 and 1. \( v_{id} \) can be limited so that \( s(v_{id}) \) does not approach too closely to 0.0 or 1.0. This ensures that there is always some chance of a bit flipping. To limit range of \( v_{id} \), a constant parameter \( v_{\text{max}} \) is often set at ±4.0, so that there is always at least a chance of \( s(v_{\text{max}}) \approx 0.018 \) that a bit will change state. In this model, \( v_{\text{max}} \) functions similarly to mutation rate in genetic algorithms (Kennedy and Eberhart, 2001).

### 1.11.2 The Particle Swarm in Continuous Numbers

The position of a particle i is assigned the algebraic vector symbol \( \overrightarrow{x_i} \). There can be any number of particles and each vector can be of any dimension. Change of position of a particle is called \( \overrightarrow{v_i} \), the velocity. Velocity is a vector of numbers that are added to the position co-ordinates in order to move the particle from one time step to another. As the system is dynamic, position of each individual is changing. The direction of movement is a function of current position and velocity, the location of individual’s previous best success called Pbest and best position found by any member in swarm is called Gbest of neighbourhood. (Kennedy and Eberhart, 2001).

### 1.11.3 The Hybrid PSO

The Hybrid PSO is an optimization algorithm combining the basic PSO with the binary PSO. Basic PSO algorithm, however, has a disadvantage that it can deal with only continuous variables. The binary PSO algorithm, on the contrary, has an ability to deal with binary variables leading to true or false decisions (Kennedy and Eberhart, 2001). If basic PSO is combined with the binary PSO, a new algorithm referred to as
hybrid PSO algorithm is formed. The fundamental idea of this hybrid algorithm is that it combines both binary and real valued parameters in one search. This hybrid optimizer is used for such problems which deal with both continuous and binary variables. It simply operates on binary inputs with binary particle swarm algorithm and treat the continuous variables with real valued particle swarm. Binary variables are added to model to take the location decisions (whether or not to locate a facility at a given candidate site), and will be taken care by binary PSO logic, while the allocation decisions are obtained by the continuous PSO algorithm. The block diagram for the hybrid swarm optimizer is shown in Fig. 1.4.

![Block Diagram](image)

**Figure 1.4 Hybrid Swarm Optimizer**

### 1.12 Bi-Objective Optimization Using Multi-Objective PSO (MOPSO)

Multi-objective optimization algorithms make use of domination concept. In these algorithms, two solutions are compared on the basis of whether one dominates the other solution or not. If there are M objective functions then a solution \( x \) is said to dominate the other solution \( y \), if both the following conditions are true:

1. The solution \( x \) is not worse than \( y \) in all objectives.
2. The solution \( x \) is strictly better than \( y \) in at-least one objective.

If any of the above conditions is violated, the solution \( x \) does not dominate the solution \( y \). Since multi-objective optimization has more than one objective to be optimized simultaneously, there cannot be a single optimal solution which
simultaneously optimizes all objectives. The resulting outcome is a set of optimal solutions with a varying degree of objective values. This set of solutions is called non-dominated set. The following two conditions must be true for a non-dominated set:

1. Any two solutions of non-dominated set must be non-dominated with respect to each other.

2. Any solution not belonging to non-dominated set is dominated by at least one member of non-dominated set. Such a non-dominated set is also called as Pareto-Optimal Set.

Because total SC cost and fill rate cannot be raised at the same time, there exists a trade-off between them. This type of system clearly represents a multi-objective optimization situation and a compromise should be made. Hence the Pareto set solutions and their corresponding decision variables should be provided, from which the decision-maker can select a solution to satisfy the industrial need (Deb, 2001).

In MOPSO, each particle has to change its position as guided by two leaders Pbest and Gbest which must be selected from the updated set of non-dominated solutions stored in the archive. The main difficulty in MOPSO is to pick suitable global best (Gbest) and personal best (Pbest) to move the particles through search space to attain a good convergence and diversity along the pareto-optimal front. For selecting Pbest a method called Prandom is employed in this work, according to which a single Pbest is maintained. Pbest is replaced if new value < Pbest, otherwise, if new value is found to be mutually non-dominating with Pbest, one of the two is randomly selected to be the new Pbest (Eveson et al., 2002). In this study, method of selection of gbest is inspired by Non-Dominating Sorted GA –II (NSGA-II). To select Gbest from archive, this work makes use of crowding distance measure and niche count. (Deb, 2001). The non-dominated solutions from the last generations are kept in the archive. The archive is an external population, in which non-dominated
solutions are kept after each flight cycle. Such an archive will allow the entrance of a solution only if: (a) it is non-dominated with respect to the contents of the archive or (b) it dominates any of the solutions within the archive. (Coelho, 2002) Finding a set of Pareto-optimal trade-off solutions are possible by running the MOPSO for many generations. Niche count $n_c$ for $i^{th}$ solution is calculated using sharing function $sh(d_{ij})$ using equations (1.11) and (1.12),

$$n_c = \sum_{j=1}^{N} sh(d_{ij}) \quad \text{and} \quad \text{.....(1.11)}$$

$$sh(d) = \begin{cases} 
1 - \left( \frac{d}{\sigma_{\text{share}}} \right)^{\alpha} & \text{if } d \leq \sigma_{\text{share}} \\
0, \text{otherwise} 
\end{cases} \quad \text{.....(1.12)}$$

It provides an estimate of the extent of crowding near a solution. Here $d_{ij}$ is the distance between the $i^{th}$ and $j^{th}$ solutions. Its value is always greater than or equal to one because $sh(d_{ij})=1$. $\sigma_{\text{share}}$ is calculated imagining that the n-dimensional hypersphere of radius $r$ must be divided among $q$ optima equally. The flow chart for the MOPSO algorithm is shown in Fig. 1.5. After calculating Niche count for each solution in the archive the Gbest is selected from the archive whose Niche count is the smallest. This ensures diversity being maintained as one which is comparatively less crowded is selected (Deb, 2001).

1.13 Research Objectives and Approach

During SC network design, in one of the phases, managers must decide on location and capacity allocation for each facility which can be made jointly. The location theory literature tends to focus on developing models for determining the number of facilities and their locations, and their assignments. These decisions are evaluated based on resulting operational shipping, inventory costs and strategic
location costs. In traditional SC management, the focus of the designs of SC network is usually on single objective, minimum cost or maximum profit. But the design, planning and scheduling projects are usually involving trade-offs among different incompatible goals such as fair profit distribution among members, customer service levels, safe inventory levels, volume flexibility etc. (Chen and Lee, 2004). Hence real SCs are to be optimized simultaneously considering more than one objective.

Figure 1.5 Flow chart for new MOPSO algorithm
The main objectives of the present research work are:

1. To formulate the mathematical model for the performance analysis and optimization of integrated two stage, three stage and four stage multi-echelon SCN problems.

2. To make some studies on integration of Design (location or strategic) decisions, Procurement plan and distribution decisions (allocation and flow decisions), along with production and inventory decisions considering uncertainty in demand and lead time.

3. To develop new swarm intelligence based optimizer to optimize simultaneously two-objectives, minimizing total SC cost and maximizing fill-rate of multi echelon SC network.

4. To validate the performance of proposed new optimizer using industrial case study.

To fulfil the research objective, the research is conducted through the following approach:

1. After deciding broad area for research, extensive literature survey was conducted in the areas of Multi-objective optimization of multi-echelon SCNs and application of heuristics for Multi-objective optimization to identify the gap.

2. Mathematical model was formulated for two-echelon SCN considering plant fixed costs and variable production and inventory costs for strategic and distribution decisions.

3. For this two-echelon SCN, a Multi-Objective Non-Linear Inertia Weight (NLIW) based hybrid PSO algorithm was developed for bi-objective optimization of Total Supply Chain Cost (TSCC) and Fill Rate (Fr). It was then tested for standard problems given in literature.
4. The model was then extended to three echelon SCN, where in Economic Order Quantity (EOQ), ordering costs, holding costs were additionally considered.

5. Performance evaluation of four-echelon SCN was then carried-out wherein shortage costs and uncertainty in demand and lead time were analyzed.

6. Statistical analysis was carried out in each of above 2, 3 and 4 stages and computational effectiveness of the algorithm was determined.

7. Two case studies were conducted to validate the performance of the algorithm.

This new optimizer can act as a decision support system for location of facilities and distribution scheduling decisions in real life multi-echelon SC optimization with many players in each stage. Whenever demand changes optimizer can be fine tuned by changing very few parameters which will make the Logistics manager’s task easier.

1.14 Limitations

The major limitation of the optimization methodology used in this work is that it is heuristic based and hence an approximate method. Though this implies that the solution obtained is sub-optimal when compared to exact method, the proposed optimizer shows satisfactory performance and leads to manageable and satisfactory situation when applied to real-life applications.

1.15 Thesis Outline

To address the various problems discussed, the thesis consists of seven chapters. In Chapter 1, an overview of SC, SCM, its evolution from Logistics management, optimization, SC optimization, exact and approximate methods to solve multi-objective optimization problems is given. Then objectives of this thesis work and thesis outline are described.
In Chapter 2, a comprehensive survey of literature on location-allocation decisions in SC, SCN optimization models, multi-objective optimization methods, Particle Swarm optimization methodology is presented.

In Chapter 3, a constrained mathematical model is formulated for two stage SCN's configuration and the application of multi-objective NLIW based hybrid PSO algorithm for performance evaluation is presented.

In Chapter 4, the mathematical model is further extended for the analysis and performance evaluation of three-echelon constrained SCN architecture. This model additionally considers EOQ, ordering costs, and holding costs.

In Chapter 5, Performance evaluation of four-echelon SCN is presented considering shortage costs and uncertainty in demand and lead time. Utility of NLIW based hybrid PSO algorithm for bi-objective optimization is presented for managerial decision making.

In Chapter 6, validation of the proposed PSO algorithm is presented for bi-objective optimization using two case studies. This chapter also explains the application of the new proposed optimizer for real life data of pump manufacturing industries.

Chapter 7 concludes this research with a discussion on some possible research extensions.