CHAPTER -2

LITERATURE SURVEY

NP-Complete and NP-Hard problems are the most complex problems to solved by computers. However, heuristic techniques such as Ant Colony Optimization ACO, Particle Swarm Optimization PSO, Genetic Algorithm GA, Simulated Annealing SA are used by many researchers to solve NP-Complete and NP-Hard problems. This chapter presents a critical analysis of efforts made by many researchers to solve NPH and NPC problems using heuristic techniques.

2.1 Difficulty in Solving NP-Complete and NP-Hard Problems

The solution of Polynomial Time problems (class P) can be found using traditional algorithm which coverage to their solution in finite time [1,16,17]. NP is a class of problems whose solution can be tested to found in finite time. P is equal to NP is a question which is not solved yet [118,119].

A problem P1 is NP-Complete, if every problem in NP can be reduced to P1 in polynomial time [1]. CNF-satisfiability was the first problem that was proved NP-Complete [25,119]. Now more than 1000 problems are proved as NP-Complete [118,119]. These problems have many application areas and can be mapped to many engineering problems. If these problems solved in quick time lot of time and effort will be saved. Cook proved that CNF-satisfiability is NP-Complete [1,25]. But many problems are still left to be uncovered and classified as NP-Complete. NP-Hard are the problems whose solution cannot be checked in polynomial time [1,20]. Most of optimization problems are NP-Hard while decision problems are NP-Complete. For example, can a given graph be colored using k-color? This decision problem is NP-Complete. If an algorithm which can solve NPH and NPC problems in reasonable time get discovered, then that algorithm will be applied to solve many problems of science

and engineering. Next section discusses the literature in which NPC and NPH problems are solved using heuristic techniques.

2.2 Heuristic Techniques to Solve NP-Complete and NP-Hard Problems

The solution of NP-Hard and NP-Complete problems is not possible in reasonable time using deterministic techniques [20,22,23]. Therefore, many heuristic techniques were used in past to solve these problems such as Genetic Algorithm [31.32], Ant Colony Optimization [19,22,60,65], Simulated Annealing [48], Tabu Search [26], Particle Swarm Optimization [16,17,82,83] etc. In recent years, many researchers worked on solving NPH and NPC problems using heuristic techniques. The major contribution in solving NPC and NPH problems using heuristic technique is analyzed in detail here.

Y. Liu et al. [22] solved NP-Hard problems using Ant Colony Optimization algorithm by applying premature convergence of ACO algorithm. A system called Physarum-based pheromone matrix optimization strategy in ant colony system was developed by Y. Liu et al. Experiments were carried out to solve TSP problem and Knapsack problem using Physarum-based ACO algorithm. The results reported in [22] confirmed that Physarum-based algorithm can solve these two problems efficiently. Another researcher, M. C. Feier et al. [20] solved K-Set and Knapsack NPC problems using parallel genetic algorithm on Compute Unified Device Architecture (CUDA). A detail comparison of sequential implementation and parallel implementation of genetic algorithm to solve NPC problems was also carried out [20]. The parallel genetic algorithm speeds up the process 67 times as compared to the sequential algorithm. S. Islam et al. [21] provided highly scalable solutions for NPC problems using Compute Unified Device Architecture (CUDA). Boolean Satisfiability Problem was solved using CUDA. The solution of the NPC problem was used in graphics card making of computer systems. The CUDA architecture tested on solving Set Cover problems of the NPC. Q. Gao and X. Xu [18] investigated computational complexity classes of problems and analyzed complexity classes in detail. Q. Gao and X. Xu analyzed the time complexity and space complexity of computational problems. The analysis of

these problems motivates to use the computer to solve these types of problems. T. Hirst and D. Harel [16] put some light on infinite variant of NP-Complete classes and presented that some problems are highly undecidable in finite time whereas some NPC problems are at low levels of complexity.

D. Zuckerman [17] suggested that some problems in the NPC set are very hard to solve when more constraints are added to these problems. Jian-wu Dang et al. [23] proposed GA to solve Multiple Travelling Salesman Problem MTSP. The algorithm performed natural selection to find the solution of MTSP problem. The complexity of that algorithm was equal to the fastest sorting algorithm. The algorithm was tested by simulation and results indicate good convergence of the algorithm. Many MTSP instanced with M=5 and number of cities ranges from 10, 20, 50 and 100 cities were solved and results confirmed better performance than the other algorithms. However, more work is to be done in the hybridization of GA using neural network learning or other techniques. It was claimed that the algorithm performed better for other optimization problems also.

N. A. Neissi and M. Mazloom [83] proposed a hybrid GA to solve TSP problem using local search heuristic algorithm. 2opt and 3opt local search operators were used and the performance of 2opt and 3opt operators was compared with each other. The algorithm was implemented on some instances of TSP problem. When 2opt local search operator was used then the convergence speed was high but when the 3opt operator was used the convergence speed was less. When time factor is important then 2opt operator should be used, priority is the case of global optimum tour length factor then 3opt operator should be used. The algorithm was implemented in three instances of TSP problem. However, the applicability of the GA using hybrid local search algorithm to solve other NPH and NPC problems needs to be discovered.

Next section discusses the work of many researchers to solve NPH and NPC problems using heuristic approaches such as ACO, PSO, GA, SA etc.

2.2.1 Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO)[7] is a heuristic technique to find solution of problems. The idea is taken from ants which discover their optimum path to a food source. ACO method is based upon the combination of distributed computation, feedback and greediness to find solution of a problem. ACO algorithm simulated the behavior of ants to solve computational problems [7,19,22,110].

The ACO algorithm in computer science is inspired by ant colonies. The searching of food is a daily life activity for ants. To search food, ants traverses a large landscape in their daily life. There are so many alternate paths available for ants to reach to the food. But the shortest path is discovered by ants using some heuristic technique.

The ants communicate with each other with the help of a communication technique known as *stigmergy*. While roaming, ants spread a chemical substance called *pheromone* among their paths. When ants move to a decision point, ants check the intensity level of the *pheromone* by smelling it and move to a path where the intensity level of *pheromone* is high. When an ant passes to a path it also spread *pheromone* and increase its intensity so that other ants can follow the same path. In this way ants found the shortest path to reach to the food source. The basic ACO algorithm used to solve many computational problems is as follows:

```
ACO Algorithm
{
        While (Terminating condition does not reach)
          {
                Perform Ant Generation Activity
                Perform pheromone evaporation
                Perform daemon actions
          }
}
```

ACO algorithm is used by many researchers to solve complex problems of computer science such as Travelling Salesman Problem, Graph Coloring Problem, Graph Coloring, N-Queen Problem and many more [60,65,79,110].

The major contribution to solve NPC and NPH problems using Ant Colony Optimization algorithms or its variants is discussed here.

M. Asif and R. Baig [19] applied Greedy Ant Colony Optimization (GACO) algorithm to solve NP-Complete problems. Ant colony algorithm was modified using greedy approach and the modified ACO algorithm found the solution of NPC problems. The modified ACO algorithm using greedy approach found the acceptable solution in the beginning of the process. The ACO algorithm was optimized using greedy heuristic technique and applied on Travelling Salesman Problem. The modified greedy ACO algorithm performed better for Travelling Salesman Problem. However, the GACO algorithm [19] did not tell about its performance for other NPC problems. A solution to all NPC problems is yet to be discovered which can be applied to solve all NPC problems. H. Shi [60] proposed a solution of knapsack problem using modified ACO algorithm. The modified ACO (MACO) algorithm was first applied on TSP problem and then the algorithm was applied on 0/1 knapsack problem. Some parameters of the algorithm were changed before applying the modified ACO algorithm on TSP. The modified algorithm was implemented in four steps. In the first step the algorithm selected the initial city for ants. In second step the modified algorithm calculated the city count. In third step the pheromone level between cities was updated. In fourth step the cycle count was incremented and ants were moved to original selected cities. The modified ACO algorithm was implemented on some instances of 0/1 knapsack problem and obtained profitable results [60]. S. Samanta et al. [65] suggested Weight Lifting Algorithm (AWL) to solve the Knapsack problem. Knapsack problem solved with single objective using ant weight lifting algorithm. The behavior of ant carry food heavier than their own weight, was noticed and applied to solve 0/1 knapsack problem. Results obtained clearly showed an improvement in performance and time complexity with respect to other genetic algorithm based approach. P. Zhao et al. [79] proposed an

improved ant colony optimization algorithm for the knapsack problem. Comparing with the basic ACO, improved ACO (IACO) algorithm [79] combined inner mutation and other mutation that made it more effective and more efficient in solving the knapsack problem. Experimental results showed that the improved ACO[79] has good convergence and provide higher solution precision.

2.2.2 Simulated Annealing (SA)

Simulated annealing [48] implies heuristic technique to solve optimization problems. The idea is taken from thermodynamics in which metal anneals and cools down. The Simulated Annealing (SA) is a very simple technique and can be easily applied to solve complex optimization problems also. From a given state, SA tries to discover the move which improve the quality of the solution. If such a move is possible then adopted otherwise select a move with "badness". The probability to select a move decreases with the badness value. The objective function here is the optimization problems objective function in place of the energy of the metal. The simple SA algorithm is shown below:

Algorithm: Simulated Annealing

1. *Select a random initial state, initial temperature and rate of cooling*
2. *Select neighbor of current state*
3. *If a neighbor is better than the current state, then pick it*
4. *If no better neighbor is found, then select it anyway on the basis of the temperature*
5. *Reduce the temperature*
6. *Repeat steps from 2 to 5 until not cooled.*

Simulated Annealing was applied to solve NP-Hard and NP-Complete problems [48,101]. Mihai Chen et al. [48] proposed solution of TSP problem by combining genetic algorithm and annealing algorithm. The hybrid approach took the advantage of

genetic algorithm to find global optimum solution and took advantage of annealing algorithm to find local optimum solutions. The applied genetic algorithm follows the steps of initial population generation, calculation of the fitness of each individual of population, performing selection operation, cross over, mutation and selection of elite individuals. The cross over rate was kept 0.80 and mutation probability was taken 0.05. The annealing algorithm was performed in two steps, first one was to design initial conditions in which temperature of annealing algorithm was kept 500 and the second step sets the Metropolis Rules Judgment. The algorithm was tested on virtual instrument programming simulation environment. The results of the simulation, test the validity of the hybrid algorithm to solve TSP problem. However, the application of the algorithm to solve other NPH and NPC problems can give better understanding of the usability.

2.2.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) algorithm [86] is a heuristic algorithm which is inspired by movement of birds and fishes. It is very similar to the Genetic Algorithm. It starts from a initial population but does not have cross over and mutation operators like GA. In PSO, particles fly in the problem space. PSO algorithm is very useful to solve optimization problems such as Travelling Salesman problem, Graph Coloring and many more. The basic PSO algorithm work as follows:

Algorithm : Particle Swarm Optimization

1. *Initialize the swarm and generate an initial population of the problem space*
2. *Calculate fitness value of each particle*
3. *Update the individual and best solution*
4. *Update velocity and position of each particle.*
5. Go to step 2 to until termination condition does not reach

PSO algorithm was used by many researchers to solve various problems NP-Hard and NP-Complete problems [86,87,95]. The major contribution of researchers to solve NPH and NPC problems using PSO is given below.

M. Muslin and M. Anantathanavit [47] proposed a hybrid algorithm using K-Mean and PSO algorithm to solve TSP problem. The algorithm used divide and conquer method to solve the TSP problem, by dividing the TSPs into smaller TSPs. The hybrid algorithm used K-Mean algorithm for clustering the cities of the TSP problem. The K-mean algorithm was one of the simple and standard clustering algorithm. Here, K-mean first calculates number of clusters centroids (K) and then added new objects of the problem to the closest centroids. After adding an object in a cluster, its centroid was calculated again. In this way the K-Mean algorithm was used to create clusters of the cities of the TSP problem. Over the cluster of cities, PSO algorithm was used to solve these sub TSPs. The PSO algorithm was further modified by using mutation operator of genetic algorithm. The mutation operator of GA was used to escape the PSO from local minimum. X. Shen et al. [69] proposed a solution to solve the knapsack problem using special particle swarm optimization by random inertia weight. The dynamic adaptive mutation was applied. The experimental results showed that proposed algorithm effectively understood the phenomenon in the evolutionary process and improved the stability of particle swarm optimization. Hence, convergent velocity and precision was increased. L. Ouyang and Dongyun Wang [73] modified particle swarm optimization (PSO) algorithm and suggested PSOGA algorithm for the knapsack problem. PSOGA algorithm depend on the standard PSO algorithm to reduce the shortcomings that standard PSO traps into local optima. PSOGA has a low convergence accuracy. PSOGA the fitness was on zero when the load bearing quantity of the knapsack was exceeded. The particle's position was reinitialized when the best position of the individual particle was same as the best position of the population. This PSOGA illustrate the excellent performance to solve knapsack problem. This PSOGA algorithm provides crossover and mutation operation to avoid premature convergence. This PSOGA algorithm was used to solve the small scale problems.

2.2.4 Genetic Algorithm (GA)

Genetic algorithm is based upon theory of evolution [14]. The algorithm starts from an initial population of candidate solutions and performs its genetic operators to find the solution of a problem. Its major operators include fitness calculation, selection, cross over and mutation. The fitness function evaluates the quality of each candidate solution in the population. The selection operator selects parent to performs cross over operation. Cross over operation generates new children. Mutation operator put some accidental and random changes in the population. These steps are repeated till an optimum solution is not found or a termination condition is not reached. The following algorithm illustrates the basic steps of genetic algorithm.

Algorithm: Genetic Algorithm

1. *Generate an initial population of candidate solutions*
2. *Calculate fitness value of each individual of population*
3. *Perform selection*
4. *Perform cross over*
5. *Perform mutation*
6. *If optimum solution found or termination condition reached then stop otherwise repeat steps 2-5*

The steps of GA are illustrated in chapter 3 in detail. Many researchers applied GA to solve NPH and NPC problems. The major contributions of different researchers in this area are discussed here.

D. Liu et al. [33] used clustering technique in evolutionary algorithm to solve TSP problem. The algorithm was divided into three phases. First phase divided all the cities into various groups using a clustering algorithm. In second phase these different group of cities were considered as smaller TSP problems and these smaller TSP problems were solved using evolutionary algorithms. In third phase the solution provided in second phase were connected together to make a final solution for the given TSP

problem. This technique was implemented and applied on some instances of TSP problem. However, the performance of the algorithm on larger instances of TSP problems is yet to be discovered. F. Frahadnia et al. [35] proposed a method using GA to solve TSP. The algorithm applied uniform order-base cross over, heuristic cross over and edge recombination cross over operators were compared. These operators were compared with inversion and reciprocal exchange mutations. The edge recombination cross over operator was useful to find the solution in shorter number of iterations. These operators avoid GA to stuck in local minima. The experiment results justified the better performance of new modified GA as compared to other existing algorithms to solve TSP. Y. Tan et al. [50] proposed a Bilevel Genetic Algorithm with Clustering (BLGAC) to solve large instances TSP problem. The BLGAC algorithm worked in three phases. The first phase divided the TSP instance into many clusters. The K-mean algorithm was used to create clusters of the TSP instance. In second phase genetic algorithm was used to solve these sub TSP problems i.e. cities divided into clusters. For every cluster GA was applied separately. These clusters were solved in parallel using genetic algorithm. In third phase the solution found by GA for different clusters were combined. Again a modified GA was used in third phase to combine different sub TSP solutions. The experimental results ensured the effectiveness of the BAGAC algorithm to solve larger instances of TSP problem.

M. Bozikovic et al. [53] proposed a solution to N-Queen problem using parallel genetic algorithm. The parallelization of genetic algorithm was applied and a global parallel genetic algorithm (GPGA) was proposed. 3-way tournament selection was used to perform selection and cross over operations in parallel. The results of the implementation verified that the parallelization improved the performance of GA to solve TSP problem. But when high level of parallelization was used, the results were not good and when low level of parallelization was used the results were good. The use of multiprocessor system was also required to achieve good results using GPGA because when single processor machine was the time got wasted in context switch. Q. Bai et al. [56] proposed a hybrid algorithm using Artificial Bee Colony (ABC)

algorithm and Variable Neighborhood Search (VNS) algorithm to solve TSP problem. The algorithm used three novel architectures and utilized problem specification knowledge into these three structures. It improved the search efficiency of VNS algorithm. The proposed hybrid algorithm using ABC and VNS first generated an initial solution of the TSP problem, then computed the possibility for the solution and obtain new solutions and estimated these solutions. Further, the identified solutions need to be replaced by the new solutions. This ABC and VNS based approach efficiently solved the TSP problem as compared to other algorithms. C. Sachdeva and S. Goel [59] proposed an improved algorithm to solve knapsack problem using GA. The algorithm solved the 0/1 knapsack problem in polynomial time. Genetic Algorithm was applied to solve knapsack and the algorithm found the solution in polynomial time. Before adding the newly generated chromosomes in the population, the new modified algorithm checked the fitness value of the newly generated chromosomes. If the fitness value of new chromosomes was good, only then these chromosomes were added in the population, otherwise rejected or discarded. In this way the modified approach improves the fitness of the population generation by generation. The modified GA found the solution in polynomial time. Experimental results ensured that the performance of modified GA was better as compared to other existing algorithms. However, it was suggested that the learning capabilities may further improve the performance of GA to solve NP-Complete problems. J. Zhao et al. [62] proposed a mathematical model based on knapsack problem. A genetic algorithm was given to solve this problem, the accuracy of solution was improved by combining the greedy strategies with traditional algorithm and it also shortened the time to solve it. Greedy strategy was analyzed in seven steps for obtaining the optimal solution. Some reference value was observed for solving knapsack problem as it reduced the Diego generation time and completely traversed the search space in finding optimal solution. P. G. Tharanipriya and P. Vishnuraja [67] proposed hybrid genetic algorithm which comprised of multi clustering genetic algorithm and rough set theory. Clustering techniques applied to get the best solution. To improve the efficiency and to get the optimal solution, multi clustering genetic algorithm was proposed. Rough set theory

used for selecting chromosomes for further process which is applied in knapsack problem [67].

Kangshun Li et al. [71] suggested a corresponding mathematical model of knapsack problem. This algorithm leads the search direction of the population and collected the result to a schema. It helped to improve the searching efficiency. At last the performance to solve knapsack problem by simple evolutionary algorithm and the evolutionary algorithm with schema was compared. Experimental results showed that an evolutionary algorithm with schema was effective to solve 0/1 knapsack problem. Y. Yang and Q. Fang [76] proposed an improved genetic algorithm (IGA) for solving Resource allocation, investment decision-making, storage allocation, loading problem and some other problems using knapsack problem. This type of genetic algorithm improved global search ability and converging velocity through contrasting the experiment results of the two algorithms. Thus the effectiveness of the improved algorithm was demonstrated. The aim of IGA was to improve the shortcomings of simple genetic algorithms. Search ability of improved algorithm was strong and the search time was also reduced, thus increased the performance of genetic algorithm. The improved algorithm was feasible and efficient.

B. H. Hasan and M. S. Mustafa [25] studied the effect of mutation operator on GA to solve Non-Deterministic Polynomial problems (NP). Different mutation operators were applied on many instances of NP problems such as TSP, Knapsack and Shubert function. The results ensured the capabilities of mutation operator in GA to avoid GA to stuck in local minimum. After applying different mutation operators on TSP, Knapsack and Shubert function, it was concluded that the performance of GA depends upon type of mutation operator used. Different mutation operators performed differently for solving TSP, Knapsack and Shubert function NP problems. The study found that the insertion mutation performed well for TSP, non-uniform mutation operator performed better for Shubert function and the performance of mutation operators remains the same for Knapsack problem.

2.3 Related Previous Work on Solving TSP Problem

Travelling salesman problem is an optimization problem. TSP problem was solved by heuristic techniques by many researchers [2,3,6,7]. This section discusses the major contribution of researchers to solve TSP problem using heuristic techniques such as ACO, GA, PSO etc. [31,32,84, 85, 86, 90, 91, 93].

Z. Lin et al. solve TSP problem using Genetic Algorithm Based on Classification (GABC) [24]. GABC algorithm classified chromosomes in different categories based upon their fitness value. It created levels of chromosomes by their fitness value. At each level GABC applied operators and generated new individuals. The fitness of these new individuals may lie in higher levels. GABC perform natural selection at each level. Thus the performance of GABC accelerated to find the solution of TSP problem. The stability and convergence of GABC algorithm was much better as compared to the other algorithms but the GABC algorithms converges slowly for large instances of TSP problem. Further work is needed to apply GABC on other instances of NP-Complete and NP-Hard problems such as Knapsack, N-Queen, Set Cover etc. R. Takahashi [27] solved TSP problem using GA by modified genetic operators. A changing cross over (CXO) operator was proposed to solve TSP using GA. The CXO operator was applied on TSP instance having more than 200 cities. The experimental results indicated that the CXO operator selects an improved EX in the starting iterations of GA and select SXX after executing several iterations. The results of GA were improved by applying CXO operator. However, GA with CXO operator was not applied on many other instances of TSP problem to check the applicability on various operators. Also CXO was not applied on other NPC and NPH problems. K. Ghoseiri and H. Sarhadi [28] applied local search in GA (GLS algorithm) to solve TSP problem. The combination of GA with local search using 2opt-DPX operator was proposed. The proposed algorithm was implemented on symmetric instances of TSP. The main idea of the approach was to first create a population using local search and the population was a rich set of chromosomes with local optimum solutions. Then GA was applied to find global optimum solutions for the TSP problem. Thus, the work combined local search

and GA to solve TSP. 2opt local optimizer was used because it was easy to use and effective on its performance. The experimental results indicated that the GLS algorithm performed very good. The performance of the algorithm can be further optimized by using hybrid genetic operators such as selection, cross over and mutation. L. Wang et al. [30] proposed an improved GA to solve TSP problem. An untwist operator was proposed which can explore the route more precisely and can shorten the length of the route. The convergence of GA using untwist operator was very high. The proposed untwist operator was applied on many instances of TSP problem and the results indicated that the untwist operator is very effective to enhance the convergence speed of the GA. However, the algorithm was not implemented on large instances of TSP problem (more than 150 cities). J. Li et al. [31] proposed a genetic differential evolution (GDE) algorithm to solve TSP problem. In GDE the differential evolution DE method was combined with genetic operators mainly cross over and mutation to solve TSP problem. The Greedy Subtour Cross Over Operator (GSX) was used to perform the cross over operation. For mutation operation the Modified Ordered (MO) operator was used. The 2-opt local search algorithm was also used to improve the local search performance. The experiments were performed on many TSP instances having 52 cities, 100 cities and 200 cities. The result indicated the effectiveness and robustness of GDE algorithm and was very good as compared to other algorithms. However, an exhaustive study about the performance of GDE algorithm to solve larger TSP instances is still left. O. Yugay et al. [32] proposed hybrid GA and applied it on TSP instances. Sorted population was taken to improve the performance of GA. A population of chromosomes was generated using random approach and the fitness of these chromosomes were calculated. A sorter initial population was taken by collecting only those chromosomes, which have higher values of fitness. The chromosomes with higher value of fitness were retained and other chromosomes were discarded. The benefit of taking such a sorted population was to use the greedy approach that the better parents will generate better children. Also the sorter population result in higher convergence speed. The hybrid population was implemented using object oriented programming and results suggested that the hybrid genetic algorithm using sorted

population performed better however what should be the size of population to get optimum convergence is yet to be discovered. Z. Wang et al. [34] proposed improved greedy GA (IGA) to solve TSP problem. The IGA algorithm first generated a good quality initial population and then applied hybrid GA algorithm to find solution of TSP problem. The IGA algorithm first generates an initial population with chromosomes having high fitness value. Then IGA applied self-adaptive cross over and self-adaptive mutation and consider better chromosomes to perform self-adaptive cross over and self-adaptive mutation. The IGA algorithm have better convergence then the other existing algorithms. L. Zhang et al. [36] proposed a solution for TSP using GA using new Cyclic Greedy Cross Over Operator (CGCO). The performance of the existing cross over operators to solve TSP problem was investigated. The CGCO operator randomly selects the city for cross over point and tried to select those cities which were not selected. This step improved the randomness criteria in evolutionary algorithm and because of it the cross over operation performed better. So the smaller improvements in the existing Cyclic Cross over and Greedy Cross over improved the result of these operators by considerable amount.

L. Yuan et al. [37] proposed an improved genetic algorithm to solve TSP problem using fine seed. The algorithm created a fine seed of chromosomes and this fine seed worked as an initial population. The other GA operators like cross over and mutation were applied on this population. The purpose to generate such a fine seed was to restrict the problem of premature convergence. Results of this technique ensured that the GA performed better even for large instances of TSP problem because the problem of premature convergence was eliminated in early phases. S. Wang and A. Zhao [38] proposed an improved GA to solve TSP problem. A hybrid genetic algorithm was proposed and applied to solve TSP problem. The algorithm added a search process in the standard GA. The search process, when best chromosome was found in the population then its neighborhood was also checked. It removed the problem of the local optimum. The retention policy replaced the worst chromosome by the best chromosomes and made the population better and better after every generation. This

scheme was very useful and experimental results confirmed the effectiveness of the hybrid GA as compared to the standard GA. M. Kuroda et al. [40] proposed Hybrid GA(HGA) and applied to solve larger instances of TSP problem. The HGA algorithm was based on a special cross over called Zoning Crossover (Z-Cross). The Zoning Crossover worked in three phases. In first phase it set zones according some rules in TSP problem. In second phase it found sub routes in the zone and cut edges inside and outside of the zone. In third phase it connected the sub tours of the zones to create a single tour. The performance of the Z-Crossover was tested on large scale TSP problems (having 39,603 to 104,815 cities). Experimental results confirmed that the Z-Crossover performed better than the other algorithms. However, HGA algorithm was only applied to TSP problem. Y. Yan et al. [41] proposed a mixed heuristic algorithm to solve TSP. The mix algorithm used Ant Colony Optimization ACO and Max-Min Ant System MMAS to make the mixed algorithm. The mixed algorithm performed better as compare to individual ACO or MMAS algorithms. The algorithm found excellent solution in a short time by the high convergence rate. L. Nian and Z. Jinhua [42] proposed hybrid GA to solve TSP. The algorithm used Ant algorithm to reduce the randomness of the genetic algorithm. The hybrid GA algorithm also used local search to find the optimal solution. In the first step algorithm generated the initial population. In second step the fitness of the chromosomes in the population was calculated. In third step the algorithm used ant algorithm to find the heuristic path. In fourth step GA operators such as selection, cross over and mutation were performed. Then local search process carried out. The fifth step adopt all the local search paths. The 6th step calculated the fitness of all the chromosomes in the population. The 7th step the stopping criteria was checked which either stop the algorithm or the steps were again repeated from step 2 to step 7. Algorithm performed better for TSP instances. S. Sharma and K. Gupta [43] compared the performance of GA to solve TSP using two different selection techniques namely, Roulette Wheel Selection (RWS) and Stochastic Universal Selection (SUS). The performance of GA to solve TSP for different values of cross over and mutation rates was also compared. Ordered cross over (OX) was proposed to solve TSP, from experimental results it was evaluated that the mutation

operator was very effective to enhance the performance of the GA. It was also concluded that for smaller size of TSP problems the SUS selection operator was better but for larger TSP problems, RWS selection operator outperformed SUS.

K. Jebari et al. [44] proposed a solution of TSP problem using Unsupervised Fuzzy Clustering Genetic Algorithm (UFCGA). The UFCGA algorithm solved the TSP problem in three phases. In the first phase the clustering technique was used to divide the TSP into many subtours. In the second phase the solution of smaller sub TSP problems were found. In the third phase the solutions of the sub TSPs were combined to get the final solution of the problem. The UFCGA algorithm was applied on several TSP instances such as *Lin105, A280, Lin318, Att532, Rat783, Pr1002* [87], and many more. The experimental results found that the UFCGA algorithm worked better for larger TSP instances as compared to smaller TSP instances. P. Chen [45] made some changes in the standard genetic algorithm and proposed an improved genetic algorithm to solve TSP problem. The improved GA provided a method to calculate fitness of the chromosomes of the GA for TSP problem. This fitness function ensured diversity of species. The improved GA algorithm calculated fitness of the chromosomes and select only those chromosomes in crossover which have high value of fitness. Secondly some improvements in the mutation operators were carried out using shift mutation and insertion mutation. The experimental results ensured that the new improved GA performed better in terms of computer time, number of iterations executed and optimum length found. T. Liu and M. Maeda [46] proposed a (SBDE) set based differential evolution approach for solving TSP problem using GA. The SBDE approach explored the search space of the TSP problem and found better solutions. The algorithm was implemented on many instances of TSPLIB [87] such as *Berlin52, Pr76* and *Lin105*. In differential evolution process first a random population of chromosomes was created, then the fitness of every chromosome was calculated, then cross over and mutation were performed. Then a trail vector for every chromosome was evaluated. The new trail vector replaced the existing chromosome if its fitness was more than an individual in the population. This process was repeated for a given number of

generations. The experimental results ensured that the results of SBDE algorithm were better as compared to other optimization techniques such as Ant Colony Optimization ACO, Max-Min Ant System, and Particle Swarm Optimization PSO. G. Singh et al. [57] proposed an improved cross over operator for real coded genetic algorithm (RCGA). The cross over operator was inspired by biological reproduction. This type of reproduction was very common in nature. The cross over operator used the concept of Boltzmann's Distribution (BD) to escape the GA from local optima. The value of Boltzmann's Distribution applied hill climbing moved and Metropolis Algorithm (MPA), which checked the possibility of survival of the new chromosomes before adding chromosomes in the population. The GA with modified cross over was applied on many instances of TSP problem. The experimental results were compared on the basis of convergence speed and quality of the solution found. It was found that the real coded genetic algorithm (RCGA) obtained good solution. J. Chen et al. [29] proposed an Elastic Net algorithm using Self Organization Map (SOM) to solve TSP problem. The algorithm was based on self-organization map structure and initialized using many artificial neurons. The algorithm presented a simple but effective modification to the elastic net and emphasis from global to local behavior during convergence and so allowing the net to ignore some image points. The experimental results indicated that the algorithm was reliable while solving Traveling Salesman Problem (TSP). However, J. Chen et al. did not apply the SOM algorithm on other instances of NPC and NPH problems.

V. Singh and S. Choudhary [39] proposed solution to GA using Modified Partially Mapped Cross Over (MPMX) operator. The algorithm applied MPMX operator to solve TSP using GA. The MPMX used the PMX as basic cross over operator and modified it to improve the performance of GA. The MPMX was applied on a TSP of 100 cities. The results ensured about the improved performance of MPMX over standard genetic algorithm. J. Stastný et al. [49] proposed graph based algorithm to solve TSP problem. The graph based approach was used for optimization of scheduling problems. Graph based approach used Generalized Lifelong Planning A* algorithm.

This algorithm was previously used for path planning of robots. The graph based approach used Generalized Lifelong Planning A* (GLPA*) which performed better than the standard Lifelong Planning A* algorithm. The GLPA* algorithm maintained priority queue which includes only locally inconsistent states which was not expanded. The graph based algorithm was applied on twenty-four standard instances of TSP problem. The results were compared with genetic algorithm with the graph based approach showed better results for some TSP instances whereas for other TSP instances the genetic algorithm performed better. M. Niendorf et al. [51] proposed a stability based method to solve TSP problem. The external conditions may get changed which may affect the quality of the solution found by any algorithm for TSP problem. The external condition includes weather, traffic conditions etc. The stability method provided an efficient way to handle external conditions. The stability based algorithm found the solution of the TSP problem by using heuristic approach. The k-opt heuristic approach with k-neighborhood was used to solve TSP instances. The algorithm was applied on many instances of TSP problems and provided shortest routes to the TSP problems when external factors need to be considered. However, more work is still needed to find a solution when optimum solution becomes sub optimum.

H. Razip and M. N. Zakaria [84] combined approximation algorithm and GA and applied on Set Covering Problem. The initial population was generated using approximation algorithm and then this generated initial population was applied on some benchmark instances of Set Cover NPC problem.

2.4 Related Previous Work on Solving N-Queen Problem

N-Queen problem is an optimization problem. N-Queen problem was solved using heuristic techniques by many researchers [4,5,10,11]. This section discusses the major contribution of researchers to solve N-Queen problem [88,89,94,95] using heuristic techniques such as ACO, GA, PSO [110-114] etc.

I. Martinjak and M. Golub [26] applied and investigated the performance of heuristic algorithms to solve N-Queen problem. Simulated Annealing, Tabu search and Genetic

Algorithm were applied on many instances of N-Queen problem and the complexities of these techniques were investigated. After doing the experiments it was observed that Simulated Annealing and Genetic Algorithms performed better than Tabu search for solving N-Queen problem. For larger instances of N-Queen problem the performance of these algorithms varied. I. N. da Silva et al. [52] proposed a solution to N-Queen problem based on Hopfield model. A modified Hopfield model to solve N-Queen problem was developed and the model guaranteed to converge to equilibrium points in finite time. The model was stable and converged to solution for the given N-Queen instances. Model was simulated for N-Queen instances and it solved the problems efficiently as compared to other algorithms. A. M. Turky and M. S. Ahmad [55] applied GA for solving N-Queen problem. The GA solved N-Queen problem very efficiently as compared to other approaches like classical search algorithms or linear programming methods. J. E. Aghazadeh Heris and M. A. Oskoei [58] proposed a modified GA to solve N-Queen problem. The modified algorithm applied local search algorithm to enhance the performance of GA. The minimum conflicts algorithm was used as a local search algorithm. The minimum conflict algorithm locally searched the search space to find the optimum solutions. The modified algorithm was a combination of minimal conflict algorithm and genetic algorithm. The minimum conflict algorithm generated chromosomes using such that the queens in the chromosome has minimum conflicts with each other. The minimum conflict algorithm placed the queens row by row. When a new queen was to be inserted then it tried to find a column in the new row such that the queen conflicts with minimum number of other queens. Thus, this technique efficiently generated chromosomes of the population for GA. The modified algorithm was applied on N-Queen problem and its performance in terms of convergence speed and results ensured that the new hybrid algorithm performed better than the other techniques. C. Zeng and T. Gu [54] proposed a new evolutionary algorithm (EA) to solve N-Queen problem. A novel EA was proposed and applied on N-Queen problem. The chromosomes of the EA were made up of assembly parts, seeds and information about status. Five rules for novel assembly algorithm included. The first rule was about assembly of the queens in the chromosome if there was no conflict among the queens.

The second rule was about assembly of the queens if there were conflicts between the queens. The third rule was to kill chromosomes which have deadlock status and regenerate these chromosomes using random approach. The fourth rule was about cross over and mutation operations of the evolutionary algorithm. The fifth rule was about the termination of the evolutionary algorithm. The assembly based EA was implemented and applied on many instances of the N-Queen problem. The results showed that the assembly based EA found solution of the N-Queen problem faster than the other existing algorithms.

2.5 Related Previous Work on Solving Knapsack Problem

Knapsack problem is an optimization problem. This problem is discussed in chapter 1. N Knapsack problem was solved through heuristic techniques by many researchers. This section discusses the major contribution of researchers to solve Knapsack problem using heuristic techniques such as ACO, GA, PSO etc. [96 -109].

C. Atılgan and U. Nuriyev [63] presented a new approach (HHA) for solving multidimensional knapsack problem. Some initial solutions were obtained which improved with iterative procedures later. Computational experiments were done on some standard problems for testing the efficiency of algorithm. Computational experiments show that HHA developed the optimal solutions. This algorithm produced high efficiency in terms of solutions and time. X. Xiao-hua et al. [64] proposed Competitive Decision Algorithm CDA for MKP (Multiple Knapsack Problem). CDA (competitive decision algorithm) solved complex optimization problems. The MKP defined as generalization of simple knapsack problem and classified as NP-Hard optimization problem. Natural selection process was investigated by recognizing that an entity with more resources has a high chance to survive. An algorithm CDAMKP was developed to solve MKP. For improved solutions a new technology was proposed and computational investigations were performed for high efficiency of algorithms. The experimental results got optimal value or near-optimal value of all the problems efficiently and applied in a wider range of applications. The results also indicated that

the CDAMKP algorithm performed better in terms of running times and quality of the solutions found. S. N. Mohanty and R. Satapathy [66] proposed an evolutionary algorithm for solving multi objective knapsack problem. This problem appeared in many real life worlds. For solving this problem, sequence of decisions viewed. The most effective evolutionary algorithm was proposed which was derivative free stochastic optimization methods based on the concept of natural selection and evolutionary process. Experiments were realized using the best and recent algorithm. Experimental result showed that the new proposed algorithm outperformed the existing evolutionary approach for knapsack problem. S. Mahato and S. Biswas [68] proposed a technique to find better quality solution and faster convergence using fuzzy genetic algorithm. In this, every item was monitored separately. Each chromosome was classified as a whole. The objective of technique was to maintain the diversity of the population. On the basis of classification of chromosome crossover was done. Genetic algorithm obtained faster convergence and better quality results. The crossover technique helped in getting good results. Y. Ma and J. Wan [72] suggested hybrid adaptive genetic algorithm (HAGA). HAGA was combined with greedy algorithm. HAGA illustrated a method for an improved adaptive genetic algorithm and repaired the infeasible solution with greedy algorithm. Hybrid adaptive genetic algorithm (HAGA) was different from the traditional genetic algorithm. HAGA obtained a better quality and faster solution speed. X. Guo et al. [74] proposed a new solution namely Chaotic Genetic Algorithm (CGA) to solve the classic knapsack problem. CGA provided a new idea into genetic algorithm. CGA added some disturbance to help finding better solution in the traditional genetic algorithm. Experimental results showed that it has high efficiency and good ability of global optimization. The result showed that running time was shorter for 0/1 knapsack problem by chaotic genetic algorithm. CGA was not an ideal method for large scale knapsack problem. Q. J. Li and K. Y. Szeto [75] proposed (GAMM) an adaptive genetic algorithm using mutation matrix. GAMM gave the solution of 0/1 knapsack problem of high complexity and structure. The evolution of the population was based on a time dependent mutation matrix. GAMM was guided by the locus statistics and the fitness distribution of the population.

GAMM provided a parameter free framework for the adaptive genetic algorithm. Three structure of the knapsack problem were used as the test cases: simple knapsack, parallel knapsack, and the layer knapsack. For each knapsack structure problem an index of difficulty was constructed. This index was based on the constraint and the number of item used. Experiments were performed to test these three model of knapsacks. Experimental results for different knapsack problems were discussed and heuristic explanation was given. A directed mutation improved the performance of the three model of knapsacks. Hierarchical structure in the layer knapsack problem appeared in many logistic problems. W. Xing and Z. Wenpeng [77] proposed a solution for knapsack problem using multi-object clonal GA (MOCGA). The proposed MOCGA algorithm took advantage of ICA algorithm. The MOCGA algorithm avoided premature convergence by using ICA algorithm. The cross over rate in MOCGA algorithm was taken between 0.85 and 0.95. This cross rate was producing good results. The MOCGA algorithm was implemented and applied on some instances of knapsack problem and the results outperformed some existing algorithms. S. Zou et al. [78] proposed a GA for the multi objective knapsack problem (MKP). Optimization ability of GA was improved by using MMGA which provided the cooperation between global exploration and local development. MMGA was more cooperative than NSGA II due to multi-objective multi-population capability. MMGA prevented the premature and degradation phenomena with the help of dynamic adjustment of parameters of genetic operators by using master-slave multi population cooperation and the mechanism of holding non-dominate solutions. Other multi-objective problems can be optimized with the help of MMGA. Y. Liu and C. Liu [80] proposed an algorithm for the population distribution adjustment by using schema-modified operator. The Schema-Guiding Evolutionary Algorithm (SGEA) provided an elite-schema space and cluster-center schema to guide the direction of individual's evolution. The global and local population diversity was improved by these two strategies. SGEA was more efficient than the simple genetic algorithm, greedy algorithm and many other algorithms in terms of performances, features and results. SGEA was used to solve the knapsack problem. However, the application of the SGEA algorithm to solve other NPH and NPC

problems was not discussed. H. Yoshizawa and S. Hashimoto [81] proposed an algorithm for landscape of knapsack problems. Structure oriented search algorithm (SOSA) worked at problems for global structure in the landscape and provided the effective results. It can directly move the next sampling domain to the promising domain. The algorithm mainly worked on promising domain by choosing sampling points from the search space which helps to find out good solutions. The function satisfies the structure of a promising domain. The SOSA provided the quality solution with less possibilities of errors. To apply the SOSA, parameter values should relate the nature of each problem in terms of parameter effectiveness and cost.

Q. Shu-Qu and W. Hui-Hong [61] proposed a solution of DSRIOA to deal with knapsack problem. DSRIOA strategies used to select excellence antibodies. In numerical experiments well-known dynamic algorithms were selected. In first algorithm, the procedure of dynamic stochastic ranking for the DSRIOA were defined and in second algorithm, the optimization version of DSRIOA was summarized. DSRIOA proved to be the promising in terms of converges capability. Also, the DSRIOA showed the better performance when it was compared to other algorithms of dynamic genetic.

Z. Zhijun et al. [70] proposed an algorithm to improve the searching efficiency. It proposed GARST algorithm by combining genetic algorithm (GA) and rough set theory(RST) to solve 0/1 Knapsack problem. GARST used the knowledge discovery function of rough set theory(RST) to get the important genes in GA. In GARST, knapsack problem was solved. GARST algorithm improved the quality of GA and searching efficiency. GARST analyzed the information hidden in the evolution process. It also determined the same gene strings of the individuals of higher fitness value. The main aim of GARST algorithm was to improve the searching efficiency and the quality of GA.

Many researchers solved several types of NPH and NPC problems such as TSP, N-Queen, Graph Coloring, Knapsack problem etc. using heuristic approaches. People

have solved the NPH and NPC problems but in a limited way i.e. the approaches were used for solving single type of NPH or NPC problem. None of the researchers have proposed and verified any generalized algorithm to solve several class of NPH and NPC problems. This observation motivated us to devise a generalized approach which can solve several class of NPH and NPC problems. To accomplish this goal, GA is found to be more suitable [5,6,7,46]. Next chapter discusses the GA usability, limitations, advantages etc. in brief and highlight the recent research work that optimized the performance of GA to solve NPH and NPC problems.