3.1 Introduction

Genetic algorithms are population based metaheuristics which work on a set of initial population and evolve the population towards better solutions by applying genetic operators of recombination and mutation. To suit the representation and the subsequent evolution of the graph based (tree based) problems such as constrained MSTs, GA has to be modified. Generating the initial population randomly in such cases leads to a poor GA performance. Also, the genetic operators of recombination and mutation do not work efficiently on such problems. Thus, a comprehensive hybrid GA model: \textit{G-GAT} has been developed which is customized to solve the BDMST problem. In course of development of this model, a need was realized to develop a fast construction heuristic to generate initial population for \textit{G-GAT}. This construction heuristic was designed to be more efficient than the existing popular construction heuristics for BDMST, such as OTTC, CBTC an RTC, so that efficient initial solutions can be generated for \textit{G-GAT}. A new crossover operator has also been suggested in \textit{G-GAT} which has a high heritability quotient and a high correlation coefficient which help in better preservation of parental substructures in offsprings. Further, two interpolation heuristic techniques have also been suggested in this work, which optimize the incumbent solutions by exchanging the edges of the BDMST with the more economical edges, so that the weight of the resulting tree is minimized. A novel chromosome representation technique has also been suggested efficient implementation of the crossover operator. These heuristics are based on local search and have been incorporated in \textit{G-GAT} as mutation operators. The mutation operators suggested in this work show a high locality and therefore, offer a better exploitation of the identified points in the search space. Further in this work, we have tried to implement a diversity
management mechanism that imposes a filter/cutoff point when the solutions in $G\cdot G\cdot A\cdot T$ or hybrid GAs in general, can be submitted to the more expensive local search phase. $G\cdot G\cdot A\cdot T$ is a steady-state hybrid GA model, which incorporates a construction heuristic and neighborhood based local search procedures to tackle the issues of diversity, premature convergence and generation of invalid solutions when GAs are applied to constrained MSTs.

In this chapter, we have given an overview of $G\cdot G\cdot A\cdot T$ along with the modifications that have been done to the standard GA in order to suit the needs of the tree based representation of the problem. The justification and supporting literature to propose different schemes and operators is the key theme here. A reference to the crossover operator, the diversity management scheme and the two interpolation heuristics may be found in the respective phase of $G\cdot G\cdot A\cdot T$, but a detailed explanation on them has been deferred for discussion in subsequent chapters.

3.2 Genetic Algorithm

Genetic algorithms are search procedures based on the concept of natural selection and evolution [76]. The key component of genetic algorithm is population, which is manipulated by the genetic operators of recombination and mutation to evolve towards optimization. GAs start with a set of solutions called initial population. These solutions are encoded using decision variables of the underlying problem into finite strings of alphabets of certain cardinality. In genetic terminology these encoded strings are known as chromosomes, the alphabets are known as genes and the values of genes are known as alleles. For example, in a problem such as MST, a chromosome represents a minimum spanning tree and a gene may represent an edge. Unlike the traditional optimization techniques, GAs work with the coding of parameters. Not with the parameters themselves. An objective function is used to distinguish the good solutions from poor solutions so as to evolve the population towards the optimum.

In a GA, once the problem is encoded in the form of a chromosome and the fitness function which will distinguish the promising solutions from the poor solutions has been decided, the
genetic operators of variation are applied to these population strings using the below listed steps:

1. **Initialization**: Initial population is usually generated randomly. Uniform distribution of such randomly generated points ensures that there is sufficient diversity in the initial population pool and that all the areas of the search space are adequately represented in the initial population. Such a search space can be large enough to be explored in reasonable time and hence domain-specific knowledge can be used to initialize the population strings which are often of higher quality and contribute to lowering of the total runtime required by the algorithm to reach the optimum. Selecting an appropriate encoding scheme to represent the solutions is a pre-requisite to this phase.

2. **Parent Selection**: Depending upon the arity of the crossover operator, \( n \) parents are selected from the population pool, known as mating pool. In general, there is a bias toward the fitter strings in the parent selection mechanism and hence the average fitness of the strings in the mating pool is more than the average fitness of the population pool. This difference in the average fitness of the two sets defines the amount of selection pressure exerted by the selection mechanism. Parent selection method depends on the requirement of selection pressure during the search. These methods generally exert varying amounts of selection pressure which aids the convergence of the algorithm to an optimum. In absence of a suitable selection pressure the search may become randomized and may take more time to converge. Thus, it is important to carefully choose the process of selection of parents and the amount of bias that should be present in the selection strategy.

3. **Crossover**: The main variation operator in genetic algorithms is the crossover operator, which combines the features from two or more parents to create new, possibly better solutions. In the simplest crossover, the features are randomly selected to be inserted in the offsprings, but domain knowledge can be built-in to bias the crossover operator. The extent to which the operator is biased has a bearing on the selection pressure and hence the performance of the algorithm. The
offsprings may not be identical to the parents but will instead combine parental traits in a novel manner.

4. **Mutation**: Mutation is traditionally considered a background operator in GAs due to the fact that the probability of mutation is kept very low. The reason for the low probability of this operator is the fact that mutation is allowed to make random changes in the chromosomes. This is against the basic evolutionary strategy where the good parental structures are to be copied in the parents and such random changes in the chromosomes might hinder the structured search of the space if the probability of mutation is kept high. Nonetheless, in absence of mutation, the population may lose all its diversity and may converge prematurely. The mutation operator in GA should be designed such that it has a high locality. Locality means that small steps in the search space, performed by mutation operators, cause small phenotypic changes. Strong locality allows evolutionary search to explore the phenotype space in a meaningful way. Weak locality prevents evolutionary search from a meaningful exploration of the phenotype space because then small variations will often cause drastic changes in phenotype. In the worst case, the search behaves like random search in the phenotype space, which is usually ineffective [158].

5. **Replacement**: The offsprings created after undergoing selection, recombination and mutation replace the original population. This is done so that the population can be kept constant over generations and because replacing the original population with the newly created one would push the search towards better individuals. Many replacement strategies such as elitist strategy, generational model of population replacement and steady-state model of replacement are used in GAs. These replacement policies differ in the take-over time of an individual in a finite population [46] and thus impact the exploration/exploitation balance of the GA.

6. **Termination Criteria**: Some criterion has to be defined to end the process of search. Performance of GA depends on the determination of the appropriate point in time to terminate the search. Termination criteria should be such that it avoids needless computations and prevents premature termination. Needless computations would
exhaust the efficiency of GA and premature termination would stop the execution of the algorithm before the search process runs out.

### 3.3 Genetic Algorithm – Hybridized

Genetic Algorithms are often a good choice for solving complex optimization problems having characteristics that make these problems difficult for classical optimization methods [47]. In particular, GAs have been applied successfully to tree problems, such as the degree-constraint minimum spanning tree problem (d-MSTP) [110], [155], the optimal communication spanning tree problem (OCSTP) [98],[143], or the bounded-diameter minimum spanning tree problem (BDMSTP) [105], [119]. The two key issues in a GA are exploration and exploitation. Exploitation means to make use of the already identified regions to locate more promising regions of the search space and exploration means to investigate new areas in search space in hope of better solutions. GAs are good at global exploration of the search space and can explore the search space effectively to identify the most promising regions of the search space. But, the inability to escape local optima in which search often gets trapped, is a serious drawback of the GAs. Local search techniques have been found good in exploiting the knowledge & resulting into fast convergence but they fail to generate a global view of the landscape. Memetic algorithms [37] make a balance between exploration and exploitation. These are the hybridization of GA with local search techniques so that both can benefit from the synergy. A hybrid algorithm merging the principles of Greedy Randomized Adaptive Search Procedures (GRASP) [61],[62] and iterated local search (ILS) [116] has been proposed in [133]. Hybridizations of the ILS metaheuristics with the multistart approaches have been successfully designed for a number of applications [9], [162] and they seem to be a promising search strategy.

Hybridization has been found to be effective in many problems where finding an optimal solution and proving its optimality require excessive computation time. Many researchers have combined local search based metaheuristics with exact algorithms of the operations research field [41]. For the constrained single objective MST problems,
Lagrangian relax-and-cut and hybrid methods [196]; and for the generalized MST problem a combined variable neighbourhood search with linear programming [71] have been proposed. However, for the BDMST problem, few hybridized algorithms exist, [84], which are mainly based on the idea of defining multiple neighborhoods as in VNS and switching strategically between these neighborhoods. However, much scope in the improvement of search techniques for these neighborhoods exists. There is also a lack of an appropriate and comprehensive strategy to define the criteria for this switching. This study aims to propose a comprehensive hybrid GA model for solving larger instances of the BDMST problem, and the customizations required therein to ensure effective application of GA to the tree based encoding of the problem along with proposing one fast construction heuristic and two local search heuristics for solving the moderate sized instances of BDMSTs.

3.4 Introduction to the Bounded Diameter Minimum Spanning Tree Problem

MST problem is a well studied combinatorial optimization problem, in the field of network design where problems can be modeled as graphs having nodes and edges between the nodes, and it is required to minimize the cost of edges connecting these nodes. MST can be solved in polynomial time by Kruskal’s algorithm [111] or Prim’s algorithm [150]. But including additional constraints in MST like degree constraint or leaf constraint often makes the problem NP-hard which are hard to be solved by polynomial time algorithms.

If it is required for example to find a configuration such that the maximum number of intermediate routers between two computers should not go beyond a certain limit, the MST to be found is then constrained by restricting the diameter of that MST and the problem thus formed, is known as the bounded diameter minimum spanning tree (BDMST) problem where we seek a tree spanning all nodes of the network of minimum costs where the diameter, i.e., the number of edges between any pair of nodes, is limited above by a given constant. BDMST problem can be formally defined as:

**Definition:** (Bounded Diameter Minimum Spanning Tree Problem, BDMST)

Given an undirected connected graph \( G = (V,E) \) with node set \( V \) \( (n = |V|) \) and edge set \( E \) \( (m = |E|) \) with associated costs \( c_e \geq 0, \forall e \in E \), we seek a spanning tree \( T = \)
\((V, E_T)\) with edge set \(E_T \subseteq E\) whose diameter, i.e., the longest path between any two nodes in the tree with respect to the number of edges, does not exceed a given constant \(D \geq 2\), and whose total costs \(C(T) = \sum_{e \in E_T} C_e\) are minimal [85].

The eccentricity of a node \(v \in V\) is defined as the maximum number of edges on the path between \(v\) and any other node within the tree \(T\). Thus, the diameter \(D\) is an upper bound for the eccentricity allowed in the BDMST. A node with minimum eccentricity is designated as the centre node, in case of even diameter trees and in case of odd diameter trees, two nodes connected by an edge, both of which have minimum eccentricities are marked as the centre nodes as shown in Figure 3.1 (a) and (b) respectively.

![Figure 3.1: (a) to (b) BDMST on a complete graph with D=4 and 7 respectively.](image)

The task of identifying a BDMST can also be seen as choosing an appropriate centre and building a height-restricted tree, where the unique path from this centre to any node of the tree consists of no more than \(H = \left\lfloor \frac{D}{2} \right\rfloor\) edges [85].

As already mentioned, the computation of an unconstrained MST is solvable in polynomial time. BDMST problem on the other hand, is known to be \(NP\)-hard for \(4 \leq D < n-1\) [124] (assuming unequal costs of the edges). In case the diameter bound \(D = 2\), the tree forms a star, i.e., all nodes are linked to the centre of the tree. In the case when \(D = 3\), the centre is a single edge and all remaining nodes of the graph are connected to one of its endpoints of the central edge by the more economical edge and hence the
optimal BDMST can be determined in polynomial time by evaluating all stars in time of the order \( O(n^2) \) when \( D = 2 \), respectively by iterating over all edges and connecting the remaining nodes in time \( O(m \cdot n) \) \( (D = 3) \), which is bounded above by \( O(n^3) \) for complete graphs [85]. For a detailed understanding on the special cases with \( D < 4 \) (refer [80]). For general cases of BDMST problem, where \( D \geq 4 \), the NP-completeness can be established by reduction to the exact cover by 3-sets (X3S) problem (refer [85]).

BDMSTs find application in various real world problems, but one of the major areas of application of BDMSTs is, ensuring quality of service in network design. Guaranteeing signal to noise ratio in communication networks would involve limiting the number of node on a path. Similarly, assuring maximum delay in such networks would again require restricting the number of nodes on any path in the network.

BDMST problem may also be encountered in the field of mutual exclusion algorithms as described by Raymond [160]. The number of messages required to enter a critical section contributes significantly to the overall cost of the distributed mutual exclusion algorithm. Thus, when the underlying communication infrastructure is based on a tree structure, the upper bound for these message become \( 2D \), where the diameter of the tree \( D \), it has a direct impact on the efficiency of the algorithm.

Another area of application of BDMSTs can be found in Linear Lightwave Networks. In designing a Linear Light-wave Network (LLN) an undirected multi-graph is used to represent the network, where multi-cast calls are sent from each source to multiple destinations. An algorithm in [15] decomposes this multi-graph \( G \) into edge disjoint trees forming at least one spanning tree. The aim however, is, to gain many spanning trees with a small diameter. Treating this problem as a BDMST problem will help to identify better tree decomposition in LLN.

Another application field of the BDMST problem is information retrieval, where it is required to compress correlated bit-vectors. Bookstein [26] explained how the sparse bit-vectors an be grouped in clusters and the relation between points in the cluster as well as the relation between the cluster representatives can be modeled as MST where the distance between two trees is used as the cost function. The decompression time of a given
bitmap vector therefore depends on its depth within the tree. Thus, the whole running time depends on the height of the built tree. This reduces the problem to the BDMST problem.

BDMST also finds application in fields such as vehicle routing problem [35], when dealing with ad-hoc wireless networks, where customers are to be provided vehicles such as the cost of the path covered by the vehicles or the number of vehicles to be used has to be minimized. This problem can be modeled as the Travelling salesman problem [TSP] with additional constraint of limiting the size of tour of each vehicle by a constant.

This list is by no means an exhaustive compilation of applications of BDMST problem in scientific as well as real world problems, but gives a fair idea of the interest of the academic community in solving BDMST problem to optimality. Fueled by the numerous application areas in which BDMST arises as a subproblem, many algorithms to solve the same have been given in literature, including some state-of-the-art hybrid metaheuristics. A detailed discussion on these has already been done in chapter 2. However, the limitation of most of these algorithms remains high computational complexity. In the section below, we discuss a hybrid GA for application to the Bounded Diameter Minimum Spanning Tree problem. However, in developing such a model, it has to be understood that the standard binary encoded GA along with the standard recombination and mutation operators will not perform well on the tree based problem of identifying the BDMSTs and a customization of almost all the phases and features of a genetic algorithm is required.

3.5 Customizing the Phases of Genetic Algorithm for BDMST Problem

3.5.1 Encoding Schemes for GA

Genetic Algorithms use an abstract representation (called chromosome) of the potential solution. Traditionally, GA's use a fixed length bit string to encode a single value solution. Besides, other encodings like real-valued numbers and data structures (trees, hashes, linked list, etc.) have also been proposed in literature. The search behavior of a Genetic algorithm depends on the interactions between these encodings that represent candidate solutions to the target problem.
A mapping between the points in the search space (phenotype) of the problem and instances of the chromosomes (genotype) must be defined. The representation can influence the success of GA and its convergence to the optimal solution within time limit. Choosing the proper representation is always problem dependent. For instance, Radcliffe [153] states that “if no domain specific knowledge is used in selecting an appropriate representation, the algorithm will have no opportunity to exceed the performance of an enumerative search.” Various encoding schemes such as, Binary encoding, Gray encoding, Real number encoding etc. have been used in literature to represent the problem as a genome or chromosome.

3.5.1.1 Traditional Encoding Schemes

**Binary Encoding:** Binary encoding is one of the most common type of encoding schemes, mainly because of its ease of application to a wide variety of problems. Suppose we have a knapsack of capacity $C$ and $N$ items, then we can encode this problem as follows:

Chromosome, in this case, is a string of 0s and 1s with $N$ bits, 1 bit representing the presence or absence of a particular item. The $i^{th}$ item of problem is represented by the $i^{th}$ bit in the chromosome. If the $i^{th}$ bit is 1 it indicates that the $i^{th}$ item has been selected, 0 otherwise.

The set of all such chromosomes ($2N$) is the solution space of the problem.

<table>
<thead>
<tr>
<th>Chromosome 1:</th>
<th>10110010010011101110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2:</td>
<td>11110110010011000001</td>
</tr>
</tbody>
</table>

**Figure 3.2: Example of Binary Encoding**

The example shown above has 20 items (and therefore 20 bits) with item1 selected in both chromosome 1 and 2 whereas item2 is selected in chromosome 2 but not in chromosome 1.
**Permutation Encoding:** Permutation encoding can be used in ordering problems, such as travelling salesman problem or task ordering problem. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence. Suppose there are cities and given distances between them. We have to find a sequence of cities with a minimal travelled distance.

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>1 5 3 2 6 4 7 9 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>8 5 6 7 2 3 1 4 9</td>
</tr>
</tbody>
</table>

Figure 3.3: Example of permutation Encoding

Here, encoded chromosomes describe the order of cities the salesman visits. For example, in chromosome 1, the salesman visits city-1 followed by city-5 followed by city-3 and so on.

**Tree Encoding:** In case of Genetic Programming tree encoding is followed, where every chromosome is a tree of some objects, such as functions or commands in programming language. Tree encoding is useful for evolving programs or any other structures that can be encoded in trees. The crossover and mutation can be done in a relatively easy way.

### 3.5.1.2 Representation Schemes for Spanning Trees

When applying GAs on tree based problems such as constrained MSTs, it is necessary to encode a solution (tree) such that evolutionary search operators like crossover or mutation can be applied to it. A variety of tree encodings exist in literature. The encoding can either be an indirect or direct representation of the tree. Indirect representations such as NetKeys [169], the link-and-node-biased encoding [144], determinant factorization [4], and Prufer numbers [79], usually encode a tree (phenotype) as a list of strings (genotypes) and apply standard GA operators to the genotype. A suitable genotype-phenotype mapping is then employed to reconstruct the tree (phenotype). In contrast, direct representations encode a tree as a set of edges and apply search operators directly to this set. Therefore, no
genotype-phenotype mapping is required. Examples are edge-set encoding [156] and NetDir encoding [169].

In [144] Palmer has given a new representation, the link and node biased (LNB) encoding and compared it to 4 other tree encodings. Tang et al. [186] have suggested the characteristic vector (CV) encoding that represents a tree as a list of binary values. In this case infeasible solutions can occur which have to be repaired. But this encoding shows good results only when the instance size is small [148]. Another tree representation has been proposed by Abuali et al. [4] known as determinant factorization. This representation is based on the in-degree matrix of the original graph, and each factor represents a spanning tree if the determinant corresponding to that factor is equal to one. Tests indicate that the performance of EA using this encoding is similar to the LNB encoding. In Weighted encodings, tree construction algorithms such as Prim’s or kruskal’s are used to construct a tree from an ordered list of edges, ordered according to weight, for example: weighted encoding [154], NetKey encoding [169], and LNB encoding, as well as variants of it [144]. Encodings such as Prufer numbers [79] describe a one-to-one mapping between spanning trees nodes and strings of node labels. Other encodings such as Blob code [147] and Dandelion code [146] are based on reducing the coding problem to the problem of sorting node pairs into lexicographic order. The locality of an encoding describes how well small changes in the genotype correspond to small changes in the phenotype. Because of low locality, Prufer numbers lead to low EA performance [79]. The locality of the Blob code is higher than Prufer numbers resulting in a better EA performance than Prufer numbers [104]. Paulden and Smith [146] show that a single mutation to a Dandelion code encoded tree leads to, at the most, five edge changes in the corresponding tree, whereas in the case of Prufer number encoding no fixed locality bound can be given.

The principle disadvantage of indirect encoding remains that an appropriate genotype-phenotype mapping has to be applied to meaningfully reconstruct trees from the genotype. This decoding procedure is often quite expensive and increases the overall runtime of the algorithm as this conversion from genotype to phenotype and vice versa may have to be done more than once during the execution of the EA. Besides this, the indirect
representations do not offer a low locality. As mentioned above, a small change to a genotype (on the application of an operator) may result in a phenotype (tree) which lies far from the original tree in the search space. This hinders a structured exploration of the search space and the search process tends towards randomization. This motivated us to investigate the direct encoding schemes specifically the edge set coding.

3.5.1.3 Representation of solutions in G-GAT

For direct representations, such as edge-set coded trees [155] and NetDir coding [169] although tree-specific search operators have to be developed but there is no additional mapping from the phenotype space to a different genotype space, and the locality of such encodings remains low. This leads to better performance of EAs when direct encodings are applied. The superiority of edge-set (ES) coding, particularly when the variation operators implement edge-set-based heuristics, to several other encodings of spanning trees has been established in [155]. The crossover operator employed by G-GAT has been specifically designed to have a high heritability. Under high heritability the decoder based encoding schemes lead to a quick loss in diversity and hence premature convergence. Edge-set coding, on the other hand, has been shown, to preserve population diversity quite well in high locality and high heritability situation. And this property of edge set coding has been put forward as a potential explanation for the recent finding that ES encoding can outperform encodings on spanning-tree based problems [180].

In G-GAT the edge-set encoding as given in [155] has been adopted and modified. The complexity of data structures and the ensuing complexity in manipulating the edges when stored as vertex pair \((x, y)\) increase the overall computational complexity of the GA. Therefore, instead of storing the edges in the form of vertex pair as suggested in [155], we have proposed to assign integer indices to each of the edges. The significance of this form of edge-set encoding which we name as index-edge-set encoding can be established from the following facts. Storing the edges as integers leads to chromosomes which may be linear and easy to manipulate. Such chromosomes lend themselves easily to crossover and mutation operators. Further, to reduce the time complexity of the recombination operator,
the chromosome has been kept sorted. Operations such as sorting again can be efficiently applied to this form of encoding. Also, with index-edge-set encoding it is easier to represent the undirected graph where the edges \((x, y)\) and \((y, x)\) correspond to the same edge. Here, we assign the same index to both these edges. Furthermore, using this form of encoding scheme, we have been able to generate the edge indices on-the-fly. The task of considering the edge between a random \(v_1\) and all the vertices in the spanning tree is an operation that would become extremely time intensive when this operation has to be carried out before adding each vertex to the growing tree. A lookup table in the form of a \(n \times n\) matrix would require \(\approx n^2\) lookups on an average, for addition of each edge. We instead have chosen to generate edge indices on-the-fly, given the pair of vertices, as explained in section 4.7.1.

A chromosome under this encoding scheme can hence be a linear one-dimensional vector containing a list of integers where each integer represents an edge of the tree.

### 3.5.2 Initial Population Generation

Population is the key element of genetic algorithms, and the genetic operations like crossover and mutation are tools for manipulating the population so that it evolves towards the optimal solution. In genetic algorithms the initial population is typically generated randomly. For example if binary encoding is considered, the chromosomes for an \(n\) item knapsack problem, where \(n = 10\) as given in figure 3.4 will be comprised of a string of 0’s and 1’s, where all the 10 loci have been filled randomly.

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>1 0 1 1 0 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>0 0 0 1 0 1 1 0 1</td>
</tr>
</tbody>
</table>

*Figure 3.4: A randomly generated Binary chromosome*

Such random generation of initial population may work in certain cases, like knapsack problem and assignment problem but in case of graph based problems especially the
constraint MST problems, generating the chromosome (tree) randomly proves to be an inefficient strategy. One of the major issues in generating the initial population randomly in case of graph based problems is, the generation of invalid offsprings. In randomly generating a tree, random edges have to be included in the chromosome. Such edges may introduce cycles. Further, randomly including edges in the chromosome may lead to a tree which is likely to violate the degree or diameter or other given constraints. Such invalid trees (chromosomes) have either to be discarded or repaired using costly repair operators. Dengiz et al. in [48] introduce an initial population strategy by generating a spanning tree by using Depth-first search algorithm of Hopcroft and Ullman [94], where edges are selected randomly from the cotree set (set of edges which are not yet added to the tree), if the constraint is not satisfied or violated, the tree is repaired using the repair procedure. This repair procedure results in increasing the time complexity of the initial population generation phase. The mechanism of generation of initial population in such cases should therefore ensure generation of population in the feasible region of the search space only. This not only reduces the space to be searched in the subsequent generations but also avoids the expensive repair operators or the need to discard such invalid trees. One such strategy is, to use a fast construction heuristic to generate initial population for subsequent improvement by some metaheuristic like EA, VNS etc.

In fact, Friedrich et al. [66] have shown that the (1+1) EA in which the initial population is generated uniformly randomly is not able to obtain a good approximation for bipartite graphs in expected polynomial time. They also go on to prove that a greedy approach in form of multiobjective EA is able to produce optimal solutions for such problems quickly. Hence, using a greedy procedure for generating initial population can help the GA to obtain optimal solution which otherwise would have resulted in achieving a poor approximation of the optimum.

To evaluate the performance of an algorithm, the most popular metric used is the number of target function evaluations (FE). The upper bound of FE can be calculated as \[ FE = G \cdot N, \] where \( N \) is the population size and \( G \) is the number of generations. It has been proven that genetic algorithms are Markov chains [139] and hence are expected to
converge to an equilibrium distribution independent of the initial state. But, in many applications the state space of the chain is so large that the convergence becomes very slow and the number of generations required to reach the optimum increases. In [108] it has been observed in an experimental study that in the case of random graphs, solutions obtained by approximation algorithm which may be far from optimal may be improved in a small number of iterations by evolutionary algorithms such that nearly optimal ones are obtained. This strengthens the case of using a fast construction heuristic to generate initial population for *G-GAT*.

A fast construction heuristic- \textit{DRGH} has been proposed in chapter 4, which has been used to create initial population for *G-GAT*. Heuristics like OTTC[1], CBTC[107] and RTC[156] also exist in literature. In fact in many studies RTC and CBTC have been used as initialization routines for further refinement by metaheuristics but OTTC has been shown to work well only on random instances, on Euclidian instances, the greediness misleads OTTC and leads to formation of shorter backbones and heavier stars and consequently heavy weight of the resulting trees. Also the time complexity of OTTC is $O(n^4)$, which is prohibitive. CBTC has a worst-case time complexity of $O(n^3)$ which is again not suitable for an initialization routine, further, on larger values of $N$ and tight diameter bounds, CBTC has been shown to perform poorly. As reflected by the variance of the solutions generated in different runs of RTC on same instances, RTC is quite sensitive to the initial choice of the centre. To take care of this, RTC has to be run starting from each vertex of the instance graph, making the time complexity of RTC equal to that of CBTC i.e. $O(n^3)$. The \textit{DRGH} algorithm which has been suggested in this thesis takes care of two issues:

i) Heavy dependence of the final solution of the heuristics on the initial selection of the centre. For a detailed mechanism of \textit{DRGH}, reader is referred to chapter 4.

ii) Second issue is generation of trees having shorter backbones edges and therefore rest of the vertices connect to these edges using heavier edges. \textit{DRGH} generates trees with lighter stars and longer backbone edges leading to BDMSTs with lesser weight.
Ensuring diversity in the initial population: Quasi random sequences are used to generate initial populations for genetic algorithms in [91]. Quasi random points do not imitate random points but are designed to maximally avoid each other [151]. It has been shown that the GAs starting with initial populations that have been generated using quasi random sequences performs significantly better than GAs where the initial population has been generated randomly [12], (though their advantage diminishes with problems of higher dimensions). This reflects that, for GAs to perform better, the strings in the initial population should be different from each other, i.e. diversity in the initial population is a factor in the successful performance of GAs. To ensure this diversity, a simple sequential inhibition (SSI) process has been given in [51], a new individual is accepted to enter the population only if its distance to all the existing individuals in the population is at least equal to minimum specified distance.

We have developed a procedure called Uniqueness() as a part of G-GAT. It is designed to work on inhibition processes, where an individual is explicitly prohibited to be located closer than some predefined minimum distance greater than 0 to the other individuals. This minimum distance is a tunable parameter and can be adjusted to enforce a higher diversity in the initial population.

Population seeding [174] consists of placing creatures in the initial population which will help the population to do what the user wants. The best strategy in population seeding is to place the answer to your problem in the original population. Using DRGH as initial population generator, therefore, helps in enhancing the performance of G-GAT by creating Initial population trees in the feasible region of the search space only, avoiding the creation of all initial strings or in the nearest neighborhood and far away from optimal solution, or search of solution sticking to a local solution.

3.5.3 Parent Selection

During the execution of GA, as the population evolves, a fraction of the existing population is selected to produce a new offspring. The selection operator on one hand must ensure
that the diversity of the population is maintained and on the other hand, it must prevent premature convergence, while pushing the population towards better solutions, over the generations. The most prevalent parent selection methods include roulette wheel selection, tournament selection, and ranking selection [161]. The selection strategy selects the chromosomes in the current generation that will be used to reproduce offspring such that individuals in the next generation will have even higher fitness. The selection operator has to be carefully formulated, such that, members of the population with higher fitness have a greater probability of being selected for mating. But, it should be taken care that, the worse members of the population still have a small probability of being selected so as to make the search process reach a large part of the search space, lest the search may simply converge to the nearest local optimum. In the sections that follow, we discuss the three most popular parent selection strategies.

3.5.3.1 Roulette Wheel Selection

Roulette wheel selection [92], also known as fitness proportionate selection, selects an individual for generating offsprings, based on the probability of selection \( P_i \) associated with each individual \( i \) in the population. Fitness level of an individual is used to assign the probability of selection with each individual chromosome.

\[
P_i = \frac{\text{Fitness}(i)}{\sum_{j=1}^{n} \text{Fitness}(j)}
\]

Where, \( P_i \) is the probability of selection of \( i^{th} \) individual of the population and \( n \) is the number of individuals in the population. Here, strings with a higher fitness will be duplicated with higher probability, and will be less likely eliminated. This may cause the diversity of the population to diminish over generations and may cause a premature convergence to a local optimum. A slow convergence rate may decrease the chance of premature convergence but it may not be able to find an optimal solution in a preset limited time. Another serious drawback with roulette wheel selection method is that of finding the suitable measure of fitness for the individuals as using the objective function
value does not serve the purpose; defining the scale to measure the function value is important. Methods for scaling exist, for example, Goldberg [77] but these are complex and computationally extensive procedures.

### 3.5.3.2 Rank-based selection

Rank-based selection strategy, ranks the chromosomes in the order of fitness. This process leads to some loss of information but it avoids the complexity of re-scaling which is an issue with the roulette wheel selection. This makes Rank-based selection simpler and more efficient than roulette wheel selection. Let each string in the population be ranked in the order of fitness and let $P[k]$ be the probability of selection of the $k^{th}$ ranked string, then $P[k]$ can be calculated as:

$P[k] = \alpha + \beta k$

Where $\alpha$ and $\beta$ are constants and can be fine tuned to adjust the selection pressure, because $P[k]$ denotes a probability distribution and hence

$$\sum_{k=1}^{M} (\alpha + \beta k = 1)$$

Sorting the population to assign rank-based probabilities is a time consuming operation and affects the speed of the entire selection process.

### 3.5.3.3 Tournament selection

Tournament selection is the most popular parent selection method in genetic algorithms due to its efficiency and simple implementation [77]. In the standard tournament selection, $k$ individuals are randomly sampled with replacements from the current population of size $N$ into a tournament of size $k$. Most common setting for tournament size is $k = 2$, which is known as Binary-tournament. These $k$ selected candidates compete amongst themselves and the candidate with the highest fitness from amongst these $k$ individuals wins the tournament. Size of the tournament has a bearing on the speed of convergence. We can say, without the loss of generality, that, larger the tournament size, higher is the selection
pressure. The set of strings which have been chosen to mate, by tournament selection, consists of winners. Therefore on an average, the fitness in the mating pool is higher than that in rest of the population. The level of difference in fitness between the mating pool and the population is the selection pressure, which is said to improve the fitness in subsequent generations [127].

In [197] Zhong et al. have compared proportional roulette wheel with tournament selection on seven general test functions and concluded that algorithm with the tournament selection exhibited better convergence behavior than proportional roulette wheel selection. It has also been found that tournament selection is preferred over rank-based selection because repeated tournament selection is faster than sorting the population to assign rank-based probabilities [103]. They also found that ranking and tournament selection outperformed proportional selection in terms of maintaining steady pressure toward convergence.

In $G$-$GAT$, therefore, we choose to implement k-2-tournament selection, in which binary-tournaments are carried out $k$ times. Using this strategy $k$ parents are selected to be in the mating pool in each iteration of the algorithm. Tournament selection has been used because in tournament selection strategy, each individual gets a chance to be selected thus preserving diversity. Retaining diversity assumes a greater importance in the context of $G$-$GAT$ because the initial population has been generated using $DRGH$, which gives trees closer to the optimal solutions and hence the initial population pool will consist of trees which are quite similar. Besides retaining diversity, tournament selection offers efficient time complexity, especially if implemented in parallel. Also, the rate at which the dominant individuals takeover the entire population is low in tournament selection, as compared to other selection schemes. Further, it does not require fitness scaling or sorting [77] [23]. Another reason for this choice of selection strategy in $G$-$GAT$ is that tournament selection offers the flexibility to tune the selection pressure of the highly fit individuals by changing the tournament size. Besides, we use elitism to ensure that the most fit individuals will be copied, unchanged, to the next generation and are not destroyed by crossover or mutation operators, and hence a continuous non-decreasing maximum fitness can be maintained.
3.5.4 Crossover

The traditional crossover operators in genetic algorithm such as uniform 1-pt or \( n \)-point crossovers can be disruptive since they may result in breaking up the useful building blocks that have been accumulated over several iterations. \( n \)-point crossover in this context, is even more disruptive than 1-pt operators. Spears and De Jong [45] have investigated the influence of crossover type and population size on the performance of the genetic algorithm. It was observed that disruptive crossovers like \( n \)-point crossover operator is more effective when applied to small populations, and on larger population use of less disruptive crossover operators such as 1- and 2-point crossover leads to better performance. This builds the case for a crossover operator that is mild and leads to minimum disruption of parental building blocks such as edges, which have been accumulated over generations.

Further, it is argued, that each feature of an offspring should be obtained from at least one of its parents, and crossover should preserve properties occurring in all parents. This would help the search process to exploit the successful common substructures in the parents. This property of a recombination operator, whereby, it preserves maximal parental features in the offsprings, is known as Heritability [157]. Uniform Order-Based Crossover combines random (order based) schemata taken from the parents, but it does not guarantee that schemata common to both parents are preserved. Conversely, UX wanders out of the reduced search space and wastes time in sampling less promising regions of the (overall) search space. This unconstrained wandering is only advantageous when the added precedence constraints are too restrictive. Crossover operators which try to maximize heritability have been studied in literature. These include Scanning based Crossover operators [54] and Diagonal Crossovers [56], besides others such as Edge-recombination (ER) operator [193] and Multiparent crossover operator (MPX) [128]. But all these operators break away from the conventional recombination arity of 2. Arity of crossover operator is defined as the number of parents participating in the crossover operator to produce one or more offsprings.
Multiparent crossover operators produce an offspring by recombining the features from more than 2 parents, but this arity may not be adjustable in all cases. The first instance of a multiparent crossover, in literature occurs in [176], which is known as Global recombination. Here the offspring may inherit features from more than two parents, but the number of parents is not determined. Similarly, multi-parent Gene Pool Recombination [134] and the Gene Linkage method [179] discuss the possibility of a multi parent recombination mechanism, but fail to determine the number of parents involved in their operator. The concept of a tune-able arity has been first discussed in the multirecombinant strategy [58]. But, the first examples of general purpose multiparent crossover operators having an adjustable arity are Scanning Crossover and the Diagonal Crossover, introduced in [54] and further investigated in [55].

**Scanning Crossover:**
Scanning crossover is a generalization of Uniform crossover. \( k \) parents are chosen for mating, the \( i^{th} \) gene of any one of the \( k \) parents is selected to be the \( i^{th} \) gene of the offspring. This selection process may be different in different variations of the Scanning crossovers. If the parent from which the gene is to be selected is chosen uniformly randomly, the resulting operator is known as Uniform-Scanning [U-SCAN]. If the selection of the gene to be inherited is biased towards the parent having greater fitness, the resulting operator is Fitness-based Scanning [FB-SCAN]. If the gene is selected based on its frequency i.e. number of occurrence in the selected, it is known as Occurrence-based Scanning [OB-SCAN].

**Diagonal crossover:**
Diagonal crossover is a modification of the general \( n \)-point crossover. Instead of choosing the \( n \) points in two parents for crossing over, it selects \((n-1)\) crossover points in \( n \) parents. \( n \) offsprings are created which inherit \( n \) segments occurring ‘along the diagonals’ in each of the \( n \) parents. Number of offsprings generated using Diagonal crossover is equal to the number of parents , whereas in Scanning based crossovers, a single offspring is created using \( n \) parents.
A comparison of the Scanning crossover and Diagonal crossover made in [54] concludes that no generalized view on the superiority of one operator on the other one can be established in all contexts. Scanning results in a single offspring out of the $n$ participating parents. Therefore, $M$ parents are processed for $M$ fitness evaluations, thus scanning has an advantage. On the other hand, because the generation gap in Scanning crossover based GA is one i.e. in every iteration of GA, one new individual is added to the population, whereas in case of diagonal crossover, $M$ new individuals are added to the population in the same time. A higher generation gap, puts diagonal crossover in a more advantageous position. In fact, it was proven that the highest gain in performance due to higher arity is seen when diagonal crossover is applied on NNI (Nearest Neighbor Interaction) and the gains on RNI (Random Neighbor Interaction) are relatively lesser. While on RNI, Scanning crossover operator performed better. This motivated us to design a recombination strategy which could draw on the strengths of both the above mentioned operators. The crossover operator suggested in this study, the $k$-Guided Crossover Operator $k$-GCO consults $k$ different parent strings for $k$ fitness evaluations and produces $2k$ offspring by recombining $k$ parents and thus, has a higher generation gap as well.

### 3.5.4.1 Crossover Operator in G-GAT

It is clear from the work done in [175], that the performance of GA becomes worse on more rugged landscapes and in this context, the multi-parent operators show similar behavior to the standard crossover operators [95]. In G-GAT, it is ensured that the pool of strings available to crossover operator is such that all of them are solutions which are near to the optimum. This means that these are not scattered in the search space as random strings would be. Further, strings which are far from optimum would not figure in this pool. Hence, $DRGH$ has the effect of smoothening out the landscape. Multiparent crossover operator as mentioned above is able to deliver better performance on a landscape which is less rugged. Besides, using a greater number of parents, the child creation is biased on the basis of a larger sample from the search space. Hence, we have chosen $k$-GCO to have arity higher than 2, although it is not possible, as explained in the next section, to give a fixed optimal
value of arity in general. But, it is possible to tune the arity of \(k\)-GCO depending upon the problem at hand and instance size as the optimal number of parents depends on the length of the individuals and on the alphabet, or more general, the size of the search space and quality of initial population supplied to \(k\)-GCO. It takes \(k\) parents as input and generates \(2k\) offsprings. The \(k\) least weight offspring trees then have a chance to replace \(k\) strings from the population and hence higher generational gap then the standard steady state crossover has been kept. When adjusting the arity of \(k\)-GCO, it was found that without loss of generality, the absolute gain in performance increases while the relative gain decreases when \(k\) is increased from 1 to 6.

In [118], it has been shown that the difference in the performance of different operators such as MPX and ER, is due to the difference in their ability to transmit edges from parents to offsprings. Hence, they may produce offsprings whose fitness is more correlated to their parents than the others. \(k\)-GCO has been designed to inherit maximal parental edges. In our work, we refer to an edge as a foreign edge when an edge that is introduced into an offspring does not occur in either parent. Introduction of a foreign edge represents a failure, which occurs because a partial tree has been built and none of the parents have an edge which can connect the remaining vertices in the tree without violating the constraints. The failure rate is a count of the number of foreign edges included into an offspring and has been used to evaluate the heritability quotient of \(k\)-GCO.

The work done by Chen & Smith in [30] exploits the concept that preserving the maximal common schema of parents results in better recombination strategy, compared to standard crossover. This is termed as the Commonalnty-Based Crossover Framework [32]. Here, the offspring inherits the common features of the parents. Similarly, the common features are preserved by the crossover operator employed in the CHC algorithm, where half of the differing bits are crossed at random [59]. In CHC, the aim is to identify a single solution. In our work, instead of aiming at a single solution, we seek to obtain a large number of similar offsprings which also ensure better exploitation of the identified region.

On such a population which consist of solutions near to optimal and mostly lying in the basin of attraction of a single optima (variation in the above statement, if any, may be
made irrelevant by the number of candidates belonging to a single optima). The effect of arity of the suggested crossover operator is now tested using such a population in which all the candidates lie in close neighborhood of each other.

### 3.5.4.2 Justifying the need of $k$-GCO

According to the commonality hypothesis the schemata common to the above average solution candidates are above average [31], so they must be preserved in the offspring. Crossover operators which preserve the common schemata have been suggested in literature [42], [75], [193]. Thus, it is desirable that the crossover operator should have a high heritability in context of common schemata. The schemata common to the parents must be carried over to the offspring provided parents represent above average candidates. The parents in $G$-$GAT$ are all above average solutions as they are generated by DRGH, therefore, in tune with the above argument, $G$-$GAT$ employs a multiparent crossover operator $k$-GCO, which has been so designed that the edges appearing in the parents are always preserved provided adding these edges does not violate the diameter constraint. The design structure of $k$-GCO is such that, first the edges appearing in all the $k$ parents are inherited by all the offspring, then the edges present in $(k-1)$ parents are considered and are inherited in the offspring provided these do not violate the constraints of cycle-free graph and adding the edge does not violate the diameter constraint. Next, the edges that are present in $(k-2)$ parents are considered and so on, till all the edges that are present in the parents have been considered. This ensures that the edges which are above average and appear in maximum of the parents are preserved in the offspring.

The existing crossover operators like the Uniform Order-Based Crossover and UX operator, are disruptive in nature and do not preserve the common schemata components of the parents. $k$-GCO, while preserving the parental edges, also offers the benefit of preserving constraints, thus, generating only feasible solutions. $k$-GCO maintains a data structure called **Vertex**$\_$$\text{depth}$, which is a vector maintaining the depth of each vertex from the centre. A parental edge is chosen to be added to the tree, if it does not violate the
diameter constraint. Rather than penalizing the invalid trees or penalizing the infeasible edges \( k\)-GCO takes care to generate only valid offsprings. The Constraint_handling subroutine that has been built into \( k\)-GCO, takes care that the offsprings are generated in a manner that they respect the diameter bound; this effectively reduces the search space.

Drawing in general from the Scanning crossover and particularly from the multiparent recombination operator proposed in [54], \( k\)-GCO makes sexuality a graded feature rather than a Boolean one. \( k\)-GCO treats arity as a tune-able parameter, where, the number of parents involved in the crossover can be adjusted to suit the instance size. The results of application of \( k\)-GCO on the De Jong functions show an enhanced performance as the number of parents is raised above two. No single arity value could however be suggested for all the instance types and sizes. Arity in case of \( k\)-GCO has therefore been kept as an adjustable parameter that can be tuned according to the instance type and size. Nevertheless, on all the De Jong functions we tested, arity higher than two has been found optimal. The reason that \( k\)-GCO shows improved performance when the number of parents is greater than two can be attributed to the fact, that when using a greater number of parents, the offspring is biased because of a larger sample from the search space. Coupled with the high heritability quotient of \( k\)-GCO, a larger sample from the search space biases the offspring to have a higher correlation coefficient. Correlation coefficient [181] is one of the most popular metric to investigate operator effectiveness. A higher correlation coefficient suggests better preservation of information during inheritance and a better ability to exploit the fitness landscape. Introduction of a non-parental edge in the offspring represents a failure of the crossover operator, as this is a point of introduction of randomness in the search behavior. A larger number of potentially good parents from which we can draw the edges, ensure that the pool of edges is large enough and minimum non-parental edges have to be considered. It has been seen that even with failure rate of 5%, non-parental edges may contribute significantly to the offspring fitness particularly when these randomly chosen edges can be arbitrarily large. Moreover, Eiben in [55] proves that as the epistasis (ruggedness of the landscape) grows the advantage of more parents becomes smaller. On landscapes with significantly high epistasis, the relationship between
operator arity and algorithm performance seems to diminish [55]. Because of the non random nature of creation of initial population trees, as explained in chapter 4, the search landscape is smoothened out to an extent an it is no longer as rugged as it would be if the parents on which $k$-GCO have to act were created randomly.

3.5.5 Mutation

After crossover, the incumbent solution strings are subjected to mutation. Mutation helps in re-introducing the genetic material which is not present in the initial population or is lost during the process of evolution. Mutation prevents the algorithm to be trapped in a local minimum, by applying small random manipulations to the offsprings.

Mutation has traditionally been considered as a simple search operator. The traditional mutation operator is the simplest of all and is termed as Bit-flip mutation. In Bit-flip mutation small changes are introduced into a chromosome by flipping every gene of the chromosome with a probability $I/L$, where $L$ is the length of the chromosome. In binary encoded strings/chromosomes, flipping will cause the value at a locus to toggle, i.e 0 will be flipped to 1 and vice versa. In encoding schemes with higher cardinality for example, in real value encoding if the cardinality of the encoding scheme is 5 and the set of alphabets in the schema are $A = \{2,4,6,8,10\}$ then, if bit value is 2, it may be flipped to any value from the set $(A-2)$. More complex mutation operators have been suggested for sequence based order based representation of the problem, such as TSP, where the sequence of genes in the chromosome reflects the order in which the cities are visited, such as:

**Reverse Sequence Mutation (RSM):** In the reverse sequence mutation operator, a sequence $S$ is identified from the two chromosome. The order of genes in $S$ is reversed to generate the sequence $S'$. $S'$ is inserted back into the chromosome at the previous position or at a newly generated position.

**Twors Mutation:** Twors mutation [3] is a strategy, where two genes are chosen randomly and their positions in the chromosome are exchanged.
Centre inverse mutation (CIM): In CIM [3] the chromosome is divided into two sections. All genes in each section are inverted and then inversely placed in the same section of the child.

3.5.5.1 Local search based operators

The above mentioned operators are specific to the order based representation of the problems such as TSP. In case operators such as bit-flip are applied to BDMST problem, it will result in invalid offsprings. If we try to exchange a randomly chosen edge in the tree with any other edge from the set of possible edges, it might result in a cycle or the same edge may get repeated, leaving the offspring invalid. Even if the edges already in the tree are debarred from taking part in the process, the resulting offspring may be invalid due to violation of the diameter constraint. Thus, it becomes imperative to introduce tree-specific mutation operators that discourage or prevent the generation of invalid offsprings.

The basic evolutionary algorithm as shown in the Algorithm below contains no local improvement step for the offsprings, so the solution that is generated is not certain to be even locally optima.

Algorithm: Basic Local Search (T)

Input: initial solution $T$
Output: improved solution $T_{min}$

Step 1: $T_{min} = T$

Step 2: Repeat
Select a neighboring solution $T \in N(T)$

//where $N(T)$ is the neighborhood of $T$.

Step 3: if $f(T')$ better than $f(T)$ then // where $f(T)$ is the fitness of $T$

$T_{min} \leftarrow T'$

Step 4: Until some termination criterion is met

Step 5: Return $T_{min}$
The mutated offspring may or may not be accepted depending upon the improvement strategy adopted. One of the two local improvement strategies is generally followed:

- **Best improvement:** The complete neighborhood $N(T)$ is explored and the best solution $T'$ is chosen. The time to evaluate the entire neighborhood may not be practical until an efficient neighborhood search strategy is defined. In case of very large neighborhoods, the time taken to implement the best first strategy may offset any advantage that local search may incur.

- **Next improvement:** The neighborhood $N(T)$ is searched systematically, the first solution $T'$ with $f(T')$ better than $f(T)$ is accepted. Although the resulting solution may not be the best solution the neighborhood has to offer, yet the time taken in implementing this strategy is minimal and therefore next improvement may be a lucrative option in case of large neighborhoods.

None of the above two strategies can claim superiority over the other in the general case. However, using either of the improvement strategies, depends on various parameters like the problem to be solved, the definition of the neighborhood structure, or if there exists an efficient incremental evaluation scheme. Local search needs a stopping criterion, which can be the maximal number of iterations, a time limit, or when reaching a local optimum. VNS [83], is a neighborhood based search which uses multiple neighborhoods for the problem under consideration. It is based on the observation that a local optimum with respect to one neighborhood structure is not necessarily a local optimum to another one. A similar mutation strategy is, where a new level is assigned to each gene with a probability of $1/n$, leaving the centre nodes. The centre nodes are then swapped with other nodes, selected randomly with equal probability. Five local search neighborhood structures are defined to locally improve the BDMSTs generated. To perform a local improvement there has to be present efficient algorithms to search the neighborhood.

In $G$-$GAT$, therefore, the mutation operators have been so designed that they introduce randomness in the incumbent solution by ensuring that the resulting tree is better than the original tree in the current neighborhood. In this sense, $G$-$GAT$ can be categorized as a Memetic algorithm [37]. Two mutation operators have been introduced in
G-GAT—the Arc_Ex_Mut and the Level_Ex_Mut, which are both neighborhood based local search operators. For a detailed explanation refer chapter 5. These operators not only ensure that the resulting offspring is always valid, but also reduce the search space by inhibiting those edges, inclusion of which would violate the diameter constraint. Further, the resulting mutant is also guaranteed to be locally optimal to at least one of the two neighborhoods. However, as in case of any local search algorithm, a balance has to be found between diversification and intensification to avoid a bad convergence behavior. The key issues considered in the design of mutation operators in G-GAT are the frequency of calls made to the mutation operators, duration of search, probability of selection of each of the operators etc.

3.5.5.2 Insight into design issues of Mutation operators of G-GAT

i. Frequency of Local Search

Frequency of local search refers to the number of iterations that a genetic algorithm performs before calling the local search operator. In the traditional memetic algorithm, the frequency of local search is set to 1. However, we do come across studies where higher frequency of local search is used for example in [120] Mathias and Whitely have used a local search frequency of 2 to solve TSP, in [121] the GA was made to run for ten uninterrupted iterations before applying local search. Thus, no general view regarding the optimal frequency of local search is there in literature. The optimal frequency of local search is problem dependent and varies with the time elapsed during the execution of the algorithm because the optimal time that should be spent on local and global search in an algorithm depends on the state of population distribution at that time.

In order to fine-tune the solutions generated by a hybrid genetic algorithm, Syrjakow and Szczurwicka [185] gave the optimal switch point between the genetic algorithm and local search. This is important since an improper use of the expensive local search in a hybrid can waste the resources. The algorithm should be able to decide intelligently on both
methods, considering the benefits and costs of their utilization. For an effective and efficient search, an appropriate use of both methods, where both can interact to produce a synergy benefitting both is required.

**ii. Duration of Local Search**

In a GA + local search hybrid, the duration of local search impacts the balance between the global exploration conducted by genetic algorithms and local refinement done by neighborhood search procedure [166]. If the number of function evaluations is taken as the criteria for comparison, a hybrid with long local search duration will execute fewer iterations of the genetic algorithm than a hybrid with shorter local duration.

On continuous domains, the local search is stopped before reaching a local optimum when its step length becomes smaller than a predefined threshold. However, in discrete combinatorial optimization problems, a local search can be performed until a solution converges to a local optimum. But, this may cause loss of population diversity [195]. Therefore, a prudent truncation criteria needs to be defined for local search embedded within a GA. Also, in Tree based problems like BDMST, calculating the weight of a neighboring tree that which shares most of its edges with another tree whose weight is already known, is much cheaper than computing the weight of a complete tree.

**iii. Probability and Selection of Local Search operator**

In an algorithm where local search has been incorporated in a GA, the local search can be applied to either every individual in the population, or to a fraction of the total individuals. In traditional hybrid genetic algorithms, local search is applied to every individual in the population. This strategy may prove to be expensive, as subjecting every individual to local search, particularly on costly objective functions, leads to duplicity (repeated generation of same solutions) when individuals that fall in the same basin of attraction of the search space are subjected to local search. A key factor, therefore, in efficiency of memetic algorithms is the proportion of the total population which should be subjected to local
search, and the criterion based on which these individuals are chosen, has a great impact on the performance of such hybrids.

Various techniques, such as tuning, distribution-based [90], fitness-based [90] techniques, and local search potential [112], have been suggested to choose the optimal fraction of the individuals that should be subjected to local search. These techniques differ in their strategy to choose the individuals for local search so that the unnecessary expenditure of resources on redundant local searches can be avoided.

The above issues have been considered in designing an optimal diversity management strategy in $G$-$GAT$. In the following section we give a sketch of the diversity management scheme used in $G$-$GAT$ along with the switching criteria when the search shall jump from global to local search and vice-versa, and the benefits of using the scheme.

### 3.5.6 Diversity Management Scheme in $G$-$GAT$

Population diversity is undoubtedly a key issue in the performance of EAs. In the absence of suitable selection pressure, the search behavior tends to become randomized and no bias is exhibited in favor of promising regions of the search space on one hand and on the other hand, a high selection pressure leads to lack of diversity in the population. Diversity measures the variety of objects or difference between objects compared. Lack of diversity hinders the ability of genetic algorithm to carry out useful exploration of the search space [114] and the search often gets trapped in local optima. This phenomenon is one of the serious reasons for the failure of GAs [59], and is known as premature convergence. In literature, a variety of metrics to evaluate diversity have been studied [50]. Diversity metrics investigated in literature measure some population features such as the individual fitness values, structures, or even the combination of the two [28]. These include diversity metrics either in phenotypic space or in genotypic space [16]. Phenotypic diversity measures the differences in phenotypes of the individuals and genotypic diversity measures the difference in the structure of the individuals. Phenotypic diversity metrics include many pair-wise and ‘column-based’ measures [44] where the difference in values of a phenotypic feature is analyzed, for example the difference in weights of edges while computing a
minimum spanning tree. Genotypic measures include entropy based measures [131] or pair-wise Hamming distance [96].

We therefore introduce a unified measure of population diversity that provides an efficient method of computing population diversity, termed as Diversity_switch(). While searching a space, it is imperative to consider the impact of size of the search space on the problem and the granularity at which the space is searched. For example, if it is expected to compute the pair-wise distance between the individuals, to check the diversity, the search space is defined by \((L \times M)\) where \(L\) is the size of the chromosome and \(M\) is the number of members in the group amongst which the Pair-wise Hamming Distance is to be calculated. Contrary to the instances from literature where \(M = P\) where \(P\) is the size of the population, we have kept \(M\) to be equal to \(2k\), where \(k\) is the number of parents involved in crossover and the value of \(k\) may be of the order of 4, 5, 6 or 7. This reduction in search space has a significant bearing on the runtime efficiency of G-GAT.

During the evolution, when the search reaches a stage where exploitation seems no longer effective, then an exploration phase is needed to diversify and prevent premature convergence. This marks a cutoff point in the search process which decides when to subject the incumbent solution to the Neighborhood Based Local Search operators. This cutoff point is embedded in the Diversity_switch() [refer section 6.3.4 for detailed explanation on design and working of Diversity_switch()]. The concept of more than one neighborhoods has been adapted from VNS [94]. In G-GAT we have defined two neighborhood structures and the search process toggles the two neighborhoods in a strategic way. This idea stems from the argument that a non promising solution on the landscape created by one neighborhood structure can be a promising solution on the landscape given by another neighborhood structure. It can be seen that the process of swapping neighborhoods (for example, in case of lack of diversity generated) corresponds to a diversification of the search. Moreover, a solution that is locally optimal with respect to a neighborhood is probably not locally optimal with respect to another neighborhood.
**3.5.7 Replacement Strategy**

A replacement strategy defines which members of the population will be replaced by the new offsprings and the generation gap $G$ defines how many individuals in the population will be replaced in a single iteration of the GA. In the generational genetic algorithm, the offsprings which are generated replace the entire populations [77]. In each iteration the entire population ($N$) is replaced. A generation gap of $N$ is thus observed in such algorithms. Whereas, in the incremental/steady state genetic algorithm is one single new member inserted into the new population at any one time.

In $G$-GAT, out of the $2k$ offsprings generated in each iteration, $k$ best offsprings are selected to replace $k$ parents, provided offsprings offer better fitness individuals to be a part of the population. Thus, in a single iteration it is possible for population to take a jump of $k$. De Jong and Sarma [46] have studied the effect of changing the generation gap $G$ on the performance of GA. A larger value of generation Gap has been shown to improve the performance of genetic algorithm [82]. Further, the negative effects like genetic drift increase in intensity when the generation Gap is less [44] and this is particularly true when the population size $N$ is small, that is, less than 100. It is observed that even with an optimal crossover rate of 0.6%, 40% of the offsprings produced are clones of their parents even with a generation gap of 1 (entire population is replaced by offsprings) [44]. Keeping in view the above arguments, we have chosen a large value of generation gap in $G$-GAT than standard steady state GA where only 1 or maximum of 2 offsprings replace the strings of the current population pool.

Out of the two standard replacement strategies - deleting the oldest and deleting the worst member of the population, we have chosen to implement the later policy because deletion of the worst individuals induces a high selective pressure, even when the parents are selected randomly [77]. $G$-GAT chooses to implement the popular combination [190] where the new individual replaces the worst individual only if the new individual is better.
3.6 Design issues considered in G-GAT

Certain design issues have to be considered when developing a hybrid genetic algorithm where local search has been embedded into genetic algorithm. These issues significantly affect the performance of the resulting hybrid if efficient strategies to combat these issues are not properly implemented. Further, these issues become a key design factor when this hybrid is to be adapted to the tree based problem such as the constrained MST problems.

**Issue 1: Generation of invalid offsprings**

One of the major issues in the application of genetic algorithm to tree based representation of the problem is that the basic genetic operators of recombination and mutation may not work optimally on the encoded problem structure frequently leading to a generation of invalid offsprings which have to be discarded or repaired using costly repair operators.

**Issue 2: Disruption of the building blocks**

Genetic operators, particularly the recombination operator leads to the disruption of the building blocks, thereby degenerating the quality of the offsprings produced by losing the good genes accumulated over several iterations. It is therefore argued that disruptiveness of the crossover operator should be minimum. The crossover operator embedded in G-GAT has been designed to ensure a high probability of desirable parent edges being inherited by the offsprings, as the edges appearing in the parents are possibly the best weight and diameter edges.

**Issue 3: Locality**

Locality is an important prerequisite to prevent evolutionary search resembling pure random search. A strong locality in the neighbourhood of global optima is essential for the success of evolutionary search [157]. The mutation operators suggested in G-GAT are local search based operators. They ensure that a small change in the genotype of a candidate (for example a single edge exchange) produces a small change in the phenotype of the resulting candidate trees.
**Issue 4: Sensitivity to problem parameters**

It has been observed that increasing the modality or relatives heights of the peaks of landscape can have a delirious effect on the performance of EAs with recombination [183]. It is argued that this makes Evolutionary algorithms sensitive to problem parameters. \( G\text{-}GAT \) is robust when applied to a variety of BDMST instance types and sizes. The low variance in the fitness value of the solutions obtained during different runs of \( G\text{-}GAT \) can be explained based on the argument that the search landscape provided to \( G\text{-}GAT \) is smoothened out by the initialization algorithm.

**Issue 5: Preventing randomization of search process**

It has been observed that the high-quality solutions are not isolated in the search space but grouped together. Therefore, better solutions can be found in a structured manner by searching around already identified good solutions. The operators and selection procedure along with a large number of very similar offsprings generated, ensure that by performing small iterated changes such as exchanging one tree edge with another feasible edge to yield a mutant which is close in search space to the original candidate or generation of almost identical offsprings each of which differs from the rest in few edges.

### 3.7 Objectives Realized

**Reduction in Computational cost:**

By generating solutions only in the feasible region of the search space, and eliminating the need of costly repair operators or penalty operators, \( G\text{-}GAT \) has been able to bring down the computational cost of identifying optimal BDMST as compared to more time consuming metaheuristic such as ACO and VNS.

**Improved Solution Quality:**

A local search algorithm’s ability to locate local optima with high accuracy complements the ability of genetic algorithms to capture a global view of the search space. Incorporating a local search algorithm also introduces an explicit refinement operator which can produce high quality solutions.
**Improved Efficiency:**

A major concern in genetic algorithm design is efficiency in terms of the time needed to reach a solution of desired quality. \textit{G-GAT} minimizes the function evaluations by working in the reduced search space.

**Robustness:**

Low disruptiveness quotient of the crossover suggested combined with its high heritability makes \textit{G-GAT} less susceptible to noise/change in the input. Further, the ruggedness of the landscape has been reduced and the peaks flattened out in the initialization phase so that the GA based optimizer is not sensitive to the input parameters.

**Coverage:**

The two neighbourhood based local search operators ensure the maximum coverage of the identified regions of the search space, by the elaborate exploitation. But it is ensured that only the promising regions of the search space are exploited thereby saving on runtime of the algorithm.

**A new Multiparent crossover operator:**

A new multiparent crossover operator has been suggested in that has an adjustable arity which can be tuned according to the problem size and type, thus making sexuality a non-Boolean feature in GAs. It offers high heritability and consequently better preservation of parental features. The concept of this operator is flexible and can be easily extended to other constraint MST problems.

**Three new Heuristic strategies proposed:**

Besides the hybrid metaheuristic \textit{G-GAT}, this work also proposes two interpolation heuristics and one construction heuristic for the BDMST problems. For the standard benchmark problems tested in this work, these three heuristics perform competitively when compared to other heuristics of their respective class.