Graph Partitioning Using Metaheuristic Techniques

Naresh Ghorpade[1], H. R. Bhapkar[2]

[1]Research Scholar, [2]Assistant Professor
Department of Mathematics, MIT School of Engineering, MITADT University, LoniKalbhor, Pune – 412201, Maharashtra, India

Abstract
The graph partitioning problem aims to partition the vertices of graph into a certain number of blocks in such a way that the edge cut is minimized and balance constraint that all blocks must be of the same weight should also be maintained. This paper is dedicated to the application of metaheuristics to the optimization of graph partitioning problem. Numerous adaptations of metaheuristics for partitioning of graphs have been proposed in last twenty years. In this paper State – of – the art methods which focuses on local as well as population-based metaheuristics are analyzed in depth.

Keywords: *Graph partitioning, Optimization technique, Swarm Intelligence*

I. Introduction

Most of the real-life problems have several solutions and occasionally an infinite number of solutions may be possible. If the problem at hand admits more than one solution, optimization can be achieved by finding the best solution of the problem in terms of some performance criterion. The graph partitioning problem (GPP) deals with the partition of vertices in a certain number of blocks in such a way that the edge cut is minimized. While partitioning graph, a balance constraint that all blocks must be of the same weight should also be maintained. Thus, optimization techniques are considered necessary for best partition with optimized cut value. This paper focuses on local search optimization techniques like Simulated Annealing, Genetic Algorithm, Tabu Search, Random Walk, Neighborhood Search, Swarm intelligence-based Ant Colony Optimization, and Particle Swarm Optimization. These optimization techniques are characterized by the use of local search method, recursively applied to the solution of the problem.

II. Optimization Techniques

Multi-objective optimization (MOO) or vector optimization is the process of optimizing systematically and simultaneously a collection of objective functions [1].
Graph partitioning is a NP-hard problem with multiple conflicting objectives such as the inter-partition relationship should be minimized while maximizing the intra-partition relationship as well as balance constraint that all blocks must be of the same weight should also be maintained. Hence graph partitioning is a multi-objective optimization problem. The optimization techniques used in graph partitioning are described below:

A. Simulated Annealing

Simulated Annealing (SA) is a standard probabilistic metaheuristic for the global optimization problem of a given function in a large search space for locating a good approximation to the global optimum. It is frequently used when the search space is discrete. The main advantage of SA is its capability of moving to states of higher energies. Simulated Annealing can be effectively used in graph partitioning to find a balanced partition which can minimize edge cut. Kirkpatrick *et al.* [2] introduced simulated annealing to solve combinatorial optimization problem. Simulated annealing is tried and tested technique, which can be simply located in space, easy to locate in place and which often generates motivating results in short programming time. Hence it is an interesting method for implementation before the use of sophisticated methods. In GPP simulated annealing is used as direct graph partitioning tool [3, 4] also in multilevel partitioning it is used as partition refinement tool [5, 6].

- ***Principle and Model of Simulated Annealing***

Simulated annealing methodology is inspired by the physical process of annealing in metallurgy. In annealing, a solid is heated to a high temperature and gradually cooled down crystallization. At high temperatures, the atoms move randomly with high kinetic energy, but during the cooling process, they have a tendency to align themselves to the minimum energy state [7]. The algorithm of simulated annealing is based on two loops called as internal loop and external loop. Iterations in an internal loop continues still the system becomes stable. Whereas as external loop reduces the temperature to simulate annealing of stable systems. The internal loop generates new state by basic alterations in previous one and then applies it to

the Metropolis acceptance rule. The best state generated by the algorithm is preserved and updated successively by internal loop.

In simulated annealing,point of optimized state space *E'* is generated within a state space from the existing point of state space *E* at each step in the algorithm. Point *E'* accepted unconditionally if it has a lower cost function than *E*. But if it has a higher cost, then it is accepted using the metropolis criterion described below. This acceptance probability is proportional to the temperature *T* of the annealing process, which is lowered steadily as the algorithm proceeds.

For *E'* belonging to state space, the probability that *E'* gets selected is given by the relation:

$$P_{s\to s'} = min\left\{1, e^{-\left(\frac{f(E')-f(E)}{T}\right)}\right\} \quad (1)$$

If *T* is high initially, then high probability of making uphill moves exists. It allows the search to fully explore the state space. Simulated Annealing will converge asymptotically to global optimum under two conditions [8]:

Homogeneous Condition: If *T* is lowered to *0* in anyway, while the length of the homogeneous sequence formed by the accepted points at each temperature is increased to an infinite length.

Inhomogeneous Condition: Irrespective of the length of these isothermal chains, the cooling schedule is chosen so that *T* approaches to 0 at a logarithmically slow rate.

In practice neither of this is possible in infinite implementations, hence polynomial time approximations are used. The quality of results and the rate of convergence are affected by the choice of cooling schedule and the length of chain at each temperature. The SA program is ended if an acceptable solution is originated or if a designated final temperature is reached. Simulated Annealing is successful in a wide range of NP-hard optimization problems.

- ***Simulated Annealing for Graph Partitioning***

Johnson *et al* [9] adapted simulated annealing for graph bipartitioning. Since simulated annealing based on the notion of moving from one state to the neighboring state, in this context two partitions are neighbors if by moving single vertex from one part to the other part we can go from one partition to the other partition. Cost function for the bipartition of graph $G = (V, E)$with two neighboring partitions for $v \in V, P_1 = (V_1, V_2)$ and $P_2 = (V_1 - \{v\}, V_2 \cup \{v\})$ is:

$$f_\alpha(P_1) = cut(P_2) + \alpha(|V_1| - |V_2|)^2 \quad (2)$$

whereα is constant. Authors have shown that this cost function bisects the graph, but balance constraint is compromised. Penalty function is the second part of cost function which allows escaping from local minimum by passing through the unauthorized state. For proper choice of α, penalty function forces the balanced partition. But the disadvantage is that it involves an inability of returning to an unacceptable state for larger graphs. Heuristic technique is used to improve the balance constraint. An adaptation of simulated annealing gives results similar to KL for smaller graphs $(100 \le |V| \le 1000)$, but execution time is longer.

Simulated annealing to graph bipartitioning is extended to *k* – partitioning of weighted graphs by C. Bichot [10]. Cost function for the *k* - partitioning of graph $G = (V, E)$ with partition $P = \{V_1, V_2, \dots, V_k\}$ is:

$f_\alpha(P) = f(P) + \alpha\left(max\sum_{i\in\{1,2,\dots,k\}} w(V_i) - mini\in 1, 2, \dots, kwVi\right)$ (3)

In case of *k* – partitioning, two partitions are neighbors if by moving single vertex from one part to the other part we can go from one partition to the other *k – 1* partitions. As a result neighborhood is very large, hence adaptation to *k* – partitioning is minimal.

Adaptation of simulated annealing to optimization of GPP is relatively easy to implement with biggest advantage of its flexibility to the acceptance of different objective functions and constraints of partitioning [10]. But these adaptations are very slow as compared with other methods. SA can be used constructively for smaller graphs with non - traditional objective functions.

B. Genetic Algorithm (GA)

Genetic Algorithms (GAs) are robust ways which can be used in search and optimization issues based on Darvin's principle of natural selection. Genetic Algorithm is one of the best optimization algorithms having great potential to deal with various problem areas like graph partitioning, image processing, and routing issues. The idea behind GA is that the combination of exceptional characteristics from different ancestors generates the better and optimized off springs that is having an improved fitness function than the ancestors. Implementing this mechanism iteratively the off springs gets more optimized, resulting into higher sustainability in the environment. The parameter set of the optimization problem is required to be coded as a finite-length string or chromosome. Population in GA is a collection of strings or a chromosome [11]. Adaptation of chromosome to the environment is evaluated using objective function. Hence the objective function in GA is called as fitness function. Basic GA is composed of three operators:

Selection – Forms population by selecting parents to reproduce chromosome at first stage, then the chromosomes generated in first stage are selected to generate population for the next stage.

Crossover - This operator requires two parents to generate several offsprings by combination of genes.
Mutation– This operator is created by constrained random modifications of one or many genes of chromosomes.
Mutation is used to explore the solution space and crossover to reach to the local optima [12]. In GA population evolves iteratively, it starts with a randomly generate initial population; a new population is generated from the existing population by selection, crossover, and mutation. Fitness function in GPP is the inverse of the objective function of minimizing edge cut. Several GA adaptations have been proposed to solve graph partitioning optimization problem [13 - 15]. In GPP number of vertices of graph represents size of chromosome, crossover operators are cut vertices and mutation operators are vertices involved in exchanges between parts.

- ***Principle and Model of Genetic Algorithm***

Cross breeding with alternative chromosome within the population occurs by giving opportunities to highly appropriate chromosomes. This breeding generates chromosome as offspring and this offspring shares some characteristics taken from every parent. By favoring the mating of the additional appropriate chromosome, the foremost promising areas of the search house are explored. Convergence to an optimal solution of the population of chromosomes depends on the structure of Genetic Algorithm.
Genetic Algorithm starts by randomly selecting an initial population *P* of *n* chromosomes. At every iteration; a new population *P'* is generated by choosing two parents from *P* with the probability of selection proportional to their fitness. Generate new offsprings by crossover from selected parents with probability P_c, and then by random mutation recombine these chromosomes with some probability $P_m (0.001 \leq P_m \leq 0.01)$. Newly generated population replaces the existing population. Termination criteria can be either the number of generations or the best fit for chromosomes or even time elapsed. For specific applications, redefine or extend crossover and mutation operators and to speed up the convergence initiate a local search at the end of each generation. Depending on the choice of implementation techniques, variations in encoding of solution space into chromosomes, the size of population, mutation and crossover rate may be observed. Genetic algorithms are better known in a variety of applications.

- ***Genetic Algorithm for Graph Partitioning***

Bui and Moon [12] exploits the potential of genetic algorithm to solve graph partitioning problem. GPP studied in [13] is a graph bisection problem with minimized cut and balanced bisection using Genetic Bisection Algorithm (GBA). The GBA carries a single mutation and crossover per iteration, which decreases the number of parameters in genetic algorithm by two. Preprocessing step of GBA modifies numbering of vertices in the initial graph *G* and new graph *G '*is generated which is reference graph for the remaining algorithm. At each iteration, two parents are selected, then from these parents crossover operator generates unique offspring and at the end mutation is applied to offspring. Swapping depends on the quality of offspring generated; if the cut is comparable to its parents then the most related parent is replaced by offspring. Otherwise, least efficient chromosome of the population is replaced. The process continues still 80% of the chromosomes have same cut value. Fitness function used in GBA is:

$$f(p) = \left[\max_{q \in pop}\left(cut(q)\right) - cut(p)\right] + \frac{1}{3}\left[\max_{q \in pop}\left(cut(q)\right) - \min_{q \in pop}\left(cut(q)\right)\right] \quad (4)$$

Partitions determined by GBA in case of ordinary graphs are of better quality but it is multiple times slower than SCOTCH. M. Cross *et al.* [140] proposed use of GA using multilevel method for k partitioning of the graph. They proposed Jostle Evolutionary [JE] adaptation of hybrid multilevel GA for optimization of GPP. It starts by creating 50 chromosomes randomly with limitation of maximum 1000 iterations. Each of 50 chromosomes generates offsprings using crossover and mutation operator. The evaluation strategy replacement operator chooses the new population for JE after offspring generation. Fitness function for this algorithm is:

$$f(p) = -Cut\left(jostle(p)\right) * balance\left(jostle(p)\right), for\ p \in population \quad (5)$$

Computation time of JE is several weeks for larger graphs.
However, it gives the best solution for graph partitioning, but it is really not computationally efficient [15].

C. Tabu Search

Tabu search is a local heuristic method based on neighborhood initially proposed by Glover and Laguna [16]. It explores the solution space by constantly replacing recent solution with best non visited neighboring solution, new solution may be less efficient. A fundamental concept in tabu search is that the intelligent search must be based on learning; the usage of flexible memory explores beyond optimality and exploits the earlier state of the search to influence its future states [17]. Tabu lists are introduced to avoid cycling of recently visited solutions and optimality crossing. In graph partitioning tabu search algorithm

utilizes two neighborhood relations S_1 and S_2 based on two different move operators. These operators exchange vertices between subsets of partition. Tabu search improves the number of current best partitions.

- ***Principle and Flow of Tabu Search***

Tabu search algorithm follows the search whenever a local optimum is encountered by permitting non-improving moves; cycling back to earlier visited solutions is prevented by the exploitation of memories, called the tabu list which reports the recent history of the search. Tabus are one of the distinctive elements of tabu search when compared with hill climbing methods. Tabu search begins iteratively with local or neighborhood search from one solution to another still selected termination criteria satisfied. Each solution s has neighborhood $N(s)$CSand solution $s' \in N(s)$ is generated from s by move. Objective function for tabu search is to minimize$f(s)$. Tabu search method permits moves which improve the current objective function value and ends when no improving solution can be established. The algorithm starts by selection of $s \in S$, then find $s' \in N(s)$ such that$f(s') < f(s)$. If no such s'found then s is the local optimum and algorithm stops. Otherwise designate $s'$to new s and repeat the process.

- ***Tabu Search for Graph Partitioning***

Roland E. *et al.* [18] introduced an adaptation of tabu search algorithm for graph partitioning. In this method all vertices having same gain are placed in partition *P* ranked as *g*. Then find a nonempty partition with the highest rank, it yields vertex with maximum gain. After each move, partition structure is updated by computing gains of selected vertex and its neighbors then transfer to the appropriate partition. Results yield by this algorithm are compared with a KL –refinement algorithm and simulated annealing, gives motivating results for 149 graphs. But unfortunately works in smaller graphs with 10 to 500 vertices. Multilevel algorithm proposed in [19] uses tabu search during partitioning and refinement process. In this method modifications are suggested in the traditional version of the KL/FM algorithm based on moving a vertex only once per round, specific type of moves (*u*, *block*) are expelled only for few iterations. Number of iterations for which move (*v*, *block*) is excluded depends on function *f* and current iteration. Non excluded vertex with highest gain is always moved. If the vertex is in block V_1, then the move (v,V_1) to the block yielding the highest gain is excluded for *f(i)* iterations means vertex cannot be placed back to the block $V_1$for *f(i)* iterations [20, 21] Advanced local search *k* – way algorithm is based on tabu search, which has been applied to graph partitioning problem [22, 23].

In this algorithm, *k* has no influence on the performance in terms of computation time. However, as *k* increases it requires larger memory. Results generated in a short running time are far better than METIS and CHACO. For the prolonged running time from one minute to one hour, the algorithm is converged towards balancing which is far better than the methods having run time in weeks.

D. Random Walks (RWs)

Random walk (RWs) was originally introduced by Karl Pearson. A random walk is mathematical formalization a path which consists of random steps in succession. Random walk serves as a fundamental model for recording stochastic activities which explains the observed behavior of the processes. Generally random walks assumed as Markov Chains [24]. For different variations of RWs in graph partitioning problem concepts like local search algorithm or path determining strategy are also used. A random walk is an iterative process which can be repeated arbitrary number of times starts with vertex *v* and then randomly selects the next vertex to visit from the set of neighbors considering transition probabilities. Diffusion is and iterative natural process in which splittable entities between neighboring vertices are exchanged still all vertices has same quantity. Diffusion is nothing but special random walk and hence both can be used for identifying dense graph regions. This method is used in graph clustering but balance constraint is neglected [25].

- ***Principle and Flow of Random Walk***

In Random Walk, the next step is selected uniformly between the neighbors of the vertex. A main weakness of RW is the existence of loops in the path while travelling from the source to the destination vertex. To prevent loops, the simplest method is to introduce memory in the RWs.

RW is based on the neighborhood search techniques. At each step a node$v' \in N(v)$ is generated from the existing node *v*. If the cost function $f(v')$for $v'$is less than$f(v)$, then the node $v'is$accepted and the search proceeds by setting $v = v'$. But if $f(v')$ is greater than$f(v)$, then the point v' is accepted with probability of accepting ascending moves. The complete family of random walk can be generated by varying parameter $P(0 \leq P \leq 1)$, from greedy search to purely random search.Greedy search will result in the search converging to local minimum and hence a small nonzero uphill probability will help in escaping such minima. But, if the uphill probability is too high, the search becomes more random and the performance may drop.

- ***Random Walk for Graph Partitioning:***

Meyerhenke [26] proposed similarity measure based on diffusion resembles to the spectral partitioning can be employed within Bubble framework, with the advantages in partitioning quality. Balancing is enforces by combining with the actual partitioning process in two different ways. First one is iterative in which diffusion load in each block is multiplied by an appropriate scalar. If an appropriate scalar cannot be determined, then the second way is adopted, in which migrating flow is computed on the quotient graph of partition. To balance the partition, flow value f_{ij} between blocks i and j indicates number of nodes to be migrated. For the best solution to migration of nodes diffusive similarity values calculated within the Bubble framework are used [27, 28]. Pellegrini combined KL/FM with diffusion to speed up previous approaches of bipartitioning in tool Scotch. These results are extended for k – way partitioning with added variations within the tools DibaP and PDibaP. In collaboration with multilevel method diffusive partition generates high quality solutions, particularly in terms of communication volume and block shape. However, faster implementation of diffusion with running time independent of k is still undetermined.

E. Fusion - Fission

The Fusion – Fission method is originated from the nuclear process. The Nuclear process generates atoms with great internal cohesion which is same as matter reforming in an optimization process. In the nature, iron is an atom with the greatest cohesion, with 56 nucleons ranging from 2 to 235 for the other atom. If the number of nucleons and sort of nucleons permits, then reorganization of nucleons of atoms generates iron atom. Graph partitioning problem correlates with nuclear process, in which objective is to find a low energy organization of parts of the graph. The vertices of the graph are nucleons and parts are atoms. In Fusion – Fission process, parts of the partition are successively split or merged.

- ***Principle and Model of Fusion - Fission***

The nucleons are ejected during Fusion or Fission process in nature. If these nucleons have high energy then they perform fissions by joining other atoms. This method is divided into two parts, Initialization and Optimization. Initialization process creates a valid initial solution, whereas optimization is combinatorial local search. Mechanism of constraint relaxation is involved in this method, to get rid of energy sinks. Fusion – Fission is an iterative process which generates parts at each step.

Multilevel algorithm is used as the local search method in Fusion – Fission algorithm. It starts with an initial partition P_k of the graph G into k parts. Divide each part P_k into $k_1$smaller parts using multilevel algorithm, it relaxes the constraint of number of parts. Partition P_{kk_1}will have $k \times k_1$ parts. Each part of the partition P_{kk_1}will be vertex of new graph G' generated from G. Subsequently, partition graph G' into k_2 parts using multilevel algorithm. If $k = k_1$, search will be intensified and if $k = k_2$ or any other number then search will be diversified. Project the new partition P' of G' onto G. Hence, new partition P_{k_2} of graph G is generated. Refinement of P_{k_2} is the last step of iteration. For refinement use Walshaw – Cross refinement algorithm. This algorithm uses load distribution method to balance the partition.

- ***Fusion Fission for Graph Partitioning***

Fusion Fission is a recent method proposed by Bichot C. E. [29, 30] for optimization of graph partitioning problem, which was originally designed to solve partitioning of European airspace. This method is divided into two parts, an initialization part and loop on the number of parts. Initialization is a two step process; in first step a sequence of numbers is created which indicates the number of parts to be created at each iteration of loop on number of parts.In the next step of initialization, initial partition of k – parts is created by partitioner. Loop on the number of parts is n times loop. First step determines required number of parts for newly created partition. Fission step divides each part of the current partition P in several parts and then in the Fusion step temporary graph G' is created in which each vertex represents one of the parts of P. Next step aims to create a new partition $k_i$from the graph G', this partition is projected onto the initial graph G. Last step is the refinement of G using Global Kernighan Lin Refinement (GKLR) algorithm. Fusion Fission method increases the efficiency in comparison with the multilevel method, but its computation time is very long.

III. Swarm Intelligence

A swarm is huge number of uniform, simple agents inter acting locally among themselves, and their environment to permit global interesting behavior to appear without central control. Swarm-based algorithms are competent to generate low cost, fast, and robust solutions to several complex problems. These algorithms are known asnature–inspired orpopulation-basedalgorithms.[31, 32]. Swarm Intelligence (SI) can consequently be known as comparatively new branch of Artificial Intelligence which is used to model the collective behavior of social swarms in nature such as ant colonies, bird flocks, and honey bees. Even though these insects or swarm individuals are fairly simple with limited cap

abilities on their own, they interact jointly with certain behavior al patterns for achieving task sessential for their survival. Swarm individuals or agents interact directly or indirectly. Waggle dance of honey bees [33] is an example of direct inter action through visual or audio contact. In direct inter action occurs when one individual changes the environment and the others respond to the new environment. In direct inter action is refer redasstigmergy, which basically means communication through the environment. Pheromone trail so fants which they depositon their way while searching for food sources is an example of indirect inter action. Thearea of research presented in this depth paper focuses on Swarm Intelligence. In next section, two of the most popular model so f swarm intelligence inspired by birds flocking behavior and ant'sstigmergic behavior are analyzed in detail.

A. Ant Colony Optimization (ACO)

Ant Colony Optimization is inspired by the for aging behavior fants. At the core of this behavior is the indirect communication between the ants with the help of chemical pheromone trails, which enables them to find short paths between the irnest and food sources. Blum [34] exploited this characteristic of real ant colonies in ACO algorithms to solve global optimization problems. Ant based solution construction, pheromone update and daemon actions are the algorithmic components involved in metaheuristic of ACO. Dorigo [35] developed the firstant colony optimization algorithm and since then numerous improvements of the ant system have been proposed. Ant colony optimization algorithm (ACO) has strong robustness as well as good dispersed calculative mechanism. ACO can be combined easily with other methods; it shows well performance in resolving the complex optimization problem. The Travelling Salesman Problem is selected as example to introduce the basic principle of ACO, and now several improvement algorithms are developed for the problem of ACO. This stochastic optimization method has been successfully applied in a number of engineering as well as real world problems. ACO algorithm imitates single ant colony which constructs solution in the form of parameters associated with problem.

- ***Principle and Model of ACO***

Ant Colony Optimization (ACO) is a computational method which iteratively optimizes a problem to pheromone with regard to a given transition probability. ACO optimizes a problem by having a updated pheromone trails and moving these ants around in the search-space according to simple mathematical formulae over the transition probability and total pheromone in the region.

At each iteration; of ACO generate global ants and calculate their fitness. Update pheromone and edge of weak regions. If fitness is improved then move local ants to better regions, otherwise select new random search direction.

Update ant pheromone and evaporate ant pheromone. The continuous ACO is based on both local and global search. Local ants have capability to move towards latent region with best solution with respect to transition probability of region *k*,

$$P_k(t) = \frac{t_k(t)}{\sum_{j=1}^{n} t_j(t)} \qquad (6)$$

where$t_k(t)$ is total pheromone at region *k* and *n* is number of global ants.

Pheromone is updated using following equation

$$t_i(t+1) = (1-r)t_i(t) \qquad (7)$$

where r is pheromone evaporation rate.

Probability of selection of region for local ants is proportional to pheromone trail.

- ***ACO for Graph Partitioning***

ACO algorithm uses pheromone trails of ants to construct solution in the form parameters associated with graph parts. Most of the ACO based graph partitioning algorithms deals with the situations where several ant colonies fight with each other in which each colony represents parts in the partition. P. Koroces*et al.* [36] initiated multi - ant colonies algorithm for graph partitioning. In this algorithm grids representing world of ants are covered by them, each grid is associated with vertex of graph. Robic*et et. al.* [37] combined this approach with multilevel method. These methods provide very good partitions but computational time is extremely longer than multilevel method. On contrary F. Comellas [38] proposed single colony ACO algorithm without multilevel composition, which is very efficient for partitioning smaller graphs.

B. Particle Swarm Optimization (PSO)

Kennedy and Eberhart [39], developed swarm intelligence model inspired by birds flocking behavior called as Particle Swarm Optimization (PSO) algorithm. The PSO has particles determined from natural swarms by combining self-experiences with social experiences using communications based on iterative computations. In PSO algorithm, a candidate solution is presented as a particle. To search out to a global optimum, it uses a collection of flying particles (changing solutions) in a search area (current and possible solutions) as well as the movement towards a promising area.PSO is a metaheuristic since it makes

hardly any assumptions about optimization of the problem and hence it can search incredibly large spaces of candidate solutions. More specifically, PSO does not use the gradient of the problem being optimized. Resemble to classic optimization methods such as gradient descent and quasi-Newton methods; PSO does not require that the optimization problem should be differentiable.PSO can therefore also be used in optimization problems which are partially irregular, noisy, change over time, etc.

- ***Principle and Model of PSO***

Particle Swarm Optimization (PSO) is a computational method which iteratively optimizes a problem to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

Consider a scenario that a group of birds is randomly searching food in an area. There is only one piece of food in the area being searched. All the birds don't know exact location of food but after each iteration they know the distance from food. Hence the best strategy to reach to the food is following the bird which is nearest to the food. PSO learns from this situation and uses it to solve the optimization problems. In PSO, each single solution is a 'bird' in the search space called as 'particle'. All the particles have fitness values which are evaluated using fitness function to be optimized and have velocities that direct the flying of the particles. The particles (solutions) fly through the problem space by following the recent optimum particles. PSO is initialized with a group of random particles and then searches for optima by updating generations. In each iteration, every particle is updated by following two "best" values. The first value is the best solution (fitness) which it has achieved so far. This value is called p_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called g_{best}. After finding the two best values, the particle updates its velocity and positions with following equations,

$$V_{id}(t+1) = V_{id}(t) + c_1 R_1\left(p_{id}(t) - x_{id}(t)\right) + c_2 R_2\left(p_{gd}(t) - x_{id}(t)\right) \quad (8)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (9)$$

where:

V_{id} – Rate of position change of i^{th} particle in d^{th}dimension and t denotes iteration count

x_{id} - Position of i^{th} particle

p_{id} – Historically best position of particle

p_{gd} - Position of swarm's global best particle

R_1 and$R_2$are two n – dimensional vectors with random numbers uniformly selected between [0, 1].

c_1 andc_2 are position constant weighting parameters called as cognitive and social parameters respectively.

Memory update is done by updatingp_{id} and p_{gd} when following condition is met.

$p_{id} = p_i$ if $f(p_i) > f(p_{id})$

$p_{gd} = g_i$if$f(g_i) > f(p_{gd})$

where, $f(x)$ is the objective function subject to maximization.

Once terminated, the algorithm reports the values of p_{gd} and$f(p_{gd})$ as its solution.

The most commonly used stopping criteria for iterations are:

1. After a fixed number of iterations (or a fixed amount of CPU time)
2. After some number of iterations without an improvement in the objective function value (the criterion used in most implementations)
3. When the objective reaches a pre-specified threshold value and thus ultimately the solution is obtained.

- ***PSO for Graph Partitioning***

A limited work is being reported in literature addressing GPP using particle swarm optimization technique. R. Green *et al.* [40] combined PSO with Breadth First Search (BFS) for partitioning large graphs. This approach is based on communication between the partitions and within the partition. Objective function is to minimize inter partition communication and maximize intra partition communication. This approach is limited to canonical graphs shows improved partitions with less computation time. Kapade et al. [41] have proposed a robust image segmentation technique, which combines discrete PSO and multilevel graph partitioning algorithm to minimize undesirable over-segmentation. Greedy graph growing partition is used, which is based on employing the region adjacency graph to improve the quality of segmentation. The performance of the proposed technique is evaluated through quantitative and qualitative validation experiments on benchmark images.

IV. Conclusion

This paper explains various optimization techniques such as Simulated Annealing, Genetic Algorithm, Tabu Search, Random Walk, Fusion Fission, Ant Colony Optimization and Particle Swarm Optimization along with their adaptation in graph partitioning. Simulated

Annealing is a general solution method where the quality of result is reasonably good but due to large neighborhoods not acceptable in *k* - partitioning. The solution time is too long for larger graphs.

Genetic Algorithm (GA) belongs to a group of evolutionary algorithm which is global search heuristic technique inspired by evolutionary biology. Optimization problems based on modifications cannot be solved using GA due to poorly known fitness function which generate bad chromosome blocks cross over.

The tabu search is a metaheuristic technique in which the selection of tabu moves generating the neighborhood of a point under search is more problem specific. Finding too many parameters can result into large number of iterations and hence will not be suitable for a large and complex graphs. Random Walk does not use memory while working with a search space and hence does not store any information about previously visited locations of the search space. Faster implementation of diffusion with running time independent of *k* is still undetermined in RW. Fusion Fission is a local search algorithm where iterative improvements may take exponential time in partitioning larger graphs.

Ant Colony Optimization is bio inspired metaheuristic motivated by foraging behavior of ants.ACO produces very good partitioning results when combined with other techniques, but the solution time is too long which ranges to hours even on the fastest computer. Particle Swarm Optimization (PSO) shares several common features with Genetic Algorithm. Both algorithms start with a class of randomly generated population, both have fitness values to evaluate the population, both update population and search for the optimum solution with random techniques. However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity and also do have a memory, which is important in algorithm. PSO need to be coupled with other heuristic method to enhance the performance and hence is very less used in graph partitioning problems.

From the study, it is observed that particle swarm optimization (PSO) seems to be comparatively better optimization technique compared to others due to its simple but efficient nature. It tries to optimize a problem by an iterative method to improve a candidate solution, with regard to given quality of measure. The PSO particles move around the search space using mathematical formulae over the particle position and velocity, movement of each particle is influenced by its own local best position and at the same time also guided towards the best known position established by other particle in the search space. This methodology over the iterations is expected to move the swarm towards the best solution.

V. References

[1] R.T Marler, "Survey of multi-objective optimization methods for engineering," Springer, 2004.

[2] Kirkpatrick, S., Gelatt, C.D. and Vecchi, P. M., "Optimization by simulated annealing," Science direct, vol. 220, pp. 671-680, 1983.

[3] H. Simon and S. Teng, "How Good is Recursive Bisection," SIAMJournal on Scientific Computing, vol.18, no.5, pp. 1436 - 1445, 1997.

[4] R. Banos and C. Gil, "Multilevel Heuristic Algorithm for Graph Partitioning," in Proceedings of the European Workshop on Evalutionary Computations in Combinatorial Optimization, pp. 143 – 153, 2003.

[5] R. Banos and C. Gil, "A Parallel Multilevel Heuristic for Graph Partitioning," in Journal of Heuristics, Vol. 10, pp. 315 - 336, 2004.

[6] C. Bichot," Metaheuristic for Graph Bisection," in Proceedings of the 10th ACM Genetic and Evalutionary Computation Conference, pp. 1801 - 1802, 2009.

[7] V.Cerny, "A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," Journal of Optimization Theory and Applications, vol. 45, no.1, pp. 41-51, 1985.

[8] Aarts and Korst, "Simulated annealing and Boltzmann machines," Wiley Publication, 1989.

[9] D. Johnson, C. Shavon, and c. Aragon, "Optimization by Simulated Annealing: An Experimental Evaluation, Part – I Graph Partitioning," Operations Research Society of America, Vol. 37, No. 6, pp. 865 – 892, 1989.

[10] C. Bichot,"A new method, the fusion fission, for the relxed k – wak graph partitioning problem and comparison with some multilevel algorithms," in Journal of Mathematical Modelling and Algorithms, Vol. 6, No. 3, pp. 319 - 344, 2007.

[11] Goldberg DE., "Genetic algorithm: search, optimization and machine learning," Addison Wesley Publishing Company, 1989.

[12] T. N. Bui and B.-R. Moon. Genetic algorithm and graph partitioning. IEEE Transactions on Computers, 45(7):841–855, 1996.

[13] H. Muhlenbein and T. Mahnig. Evolutionary optimization ¨ and the estimation of search distributions with applications to graph bipartitioning. International Journal of Approximate Reasoning, 31(3):157–192, 2002.

[14] A. J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. Journal of Global Optimization, 29(2):225–241, 2004.

[15] J. G. Martin. Spectral techniques for graph bisection in genetic algorithms. In Genetic and Evolutionary Computation Conference, pages 1249–1256, 2006.

[16] Glover, F. and Laguna, M., "Tabu search," ch.3, "Modern heuristic techniques for combinatorial problems," McGraw-Hill Publications, pp 70-150, 1995.

[17] Michel Gendreau and Jean-Yves Potvin, "Tabu Search", Handbook of metaheuristics, International Series in Operations Research and Management Science, 146, Springer Science Business Media, 2010.

[18] Rolland, E., Pirkul, H. and Glover, "Tabu search for graph partitioning," Annals of Operations Research. Vol. 63 pp.209-232, 2004.

[19] C. Walshaw, "Multilevel refinement for combinatorial optimization problems"". Annals of Operations Research, Vol l31. pp. 325-372, 2006.

[20] P. Galienier, Z. Boujbel, and M. C. Fernandes, An Efficient Memetic Algorithm for Graph Partitioning, in Annals of Operations Research, pp. 1 – 22, 2011.

[21] U, Benlic, and J. K. Hao, "An Effective Multilevel Memetic Algorithm for Balanced Graph Partitioning," in 22nd IEEE Conference on Tools with Artificial Intelligence, pp. 121 - 128, 2010.
[22] U, Benlic, and J. K. Hao, "An Effective Multilevel Tabu Search Approach for Balanced Graph Partitioning," in Tans. on Computers and Operations Research, Vol. 7, No. 38, pp. 1066 - 1075, 2011.
[23] R. Diekmann and C. Walshaw, "Shape Optimized Mesh Partitioning and Load Balancing for Parallel Adaptive FEM," in Journal on Parallel Computing, vol.12, no.26, pp. 1555 - 1581, 2000.In Proc. Internatl. Conference on Parallel and Distributed Systems (ICPADS'09), pp. 150–157. IEEE Computer Society, 2009.
[24] Henning Meyerhenke, "Beyond good shapes: Diffusion-based graph partitioning is relaxed cut optimization," In Proc. 21st International Symposium on Algorithms and Computation (ISAAC'10). Springer-Verlag, 2010.
[25] CE Bichot,"A metaheuristic based on fusion and fission for partitioning problems," Parallel and Distributed Processing Symposium, 2006. IPDPS 2006.
[26] CE Bichot,"A new method, the fusion fission, for the relaxed k-way graph partitioning problem, and comparisons with some multilevel algorithms,"Journal of Mathematical Modelling and Algorithms 6 (3), pp. 319-344, 2007
[27] B. K. Panigrahi, Y. Shi, and M.-H. Lim (eds.)," Handbook of Swarm Intelligence," Series: Adaptation, Learning, and Optimization, Vol 7, Springer-Verlag Berlin Heidelberg, 2011. ISBN 978-3-642-17389-9.
[28] C. Blum and D. Merkle (eds.). Swarm Intelligence – Introduction and Applications. Natural Computing. Springer, Berlin, 2008.
[29] L´aszl´oLov´asz and Mikl´osSimonovits," Random walks in a convex body and an improved volume algorithm,"Random Struct. Algorithms, Vol.4, No. 4, pp. 359–412, 1993.
[30] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "A min-max cut algorithm for graph partitioning and data clustering," In ICDM, pp. 107–114, 2001.
[31] H. Meyerhenke, B. Monien, S. Schamberger,"Graph Partitioning and Disturbed Diffusion," Parallel Computing, Vol. 35(10-11) pp. 544-569, 2009.
[32] Henning Meyerhenke, "Dynamic load balancing for parallel numerical simulations based on repartitioning with disturbed diffusion,"
[33] M. Belal, J. Gaber, H. El-Sayed, and A. Almojel, Swarm Intelligence, In Handbook of Bioinspired Algorithms and Applications. Series: CRC Computer & Information Science. Vol. 7. Chapman & Hall Eds, 2006. ISBN 1-58488-477-5.
[34] C. Blum, Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling, Comput. Oper. Res., Vol. 32, No. 6, pp. 1565–1591, 2005.
[35] P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo, "Estimation based ant colony optimization algorithms for the probabilistic travelling salesman problem,". Swarm Intell., 3(3), pp. 223–242, 2009.
[36] P. Korosec, J. Silc and B. Robic, "Mesh Partitioning: A Multilevel Ant-Colony-oOtimization Algorithm," in Parallel and Distributed Processing Symposium, 2003.
[37] K. Taskova, P. Korosec and J. Silc, "A Distributed Multilevel Ant Colonies Approach," Informatica, vol. 32, no. 3, pp. 307-317, 2008.
[38] Comellas, F., Sapena, E., 2006, "A Multiagent Algorithm for Graph Partitioning. Applications of Evolutionary Computing," Lecture Notes in Computer Science. Springer, 3907, pp.279-285, 2008
[39] J. Kennedy and R. Eberhart, "Particle swarm optimization," in the Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, Vol. 4, pp. 1942-1948, 1995.
[40] S. Gadde, "Graph partitioning algorithms for minimizing internode communication on a distributed system," Ph. D Thesis, 2013.
[41] S. D. Kapade, "Swarm intelligence-based graph partitioning for image segmentation," 2015.

Authors Profile

Mr. Naresh Ghorpade is working as Assistant Professor in Mathematics at Sinhgad College of Arts, Science and Commerce, Narhe, Pune, India. He has 18 years teaching experience and published 03 research papers in national and international reputed journals.

Dr. H. R. Bhapkar is working as Assistant Professor in Mathematics in MIT ADT University's MIT School of Engineering, LoniKalbhor, Pune, India. He has 18 years teaching experience and published 16 research papers in national and international reputed journals. He has authored 103 books of Mathematics to different universities of India.