Two Hybrid Meta-heuristic Approaches for Minimum Dominating Set Problem

Anupama Potluri and Alok Singh

Department of Computer and Information Sciences
University of Hyderabad, Hyderabad 500046, India
{apcs,alokcs}@uohyd.ernet.in

Abstract. Minimum dominating set, which is an NP-hard problem, finds many practical uses in diverse domains. A greedy algorithm to compute the minimum dominating set is proven to be the optimal approximate algorithm unless $P = NP$. Meta-heuristics, generally, find solutions better than simple greedy approximate algorithms as they explore the search space better without incurring the cost of an exponential algorithm. However, there are hardly any studies of application of meta-heuristic techniques for this problem. In some applications it is important to minimize the dominating set as much as possible to reduce cost and/or time to perform other operations based on the dominating set. In this paper, we propose a hybrid genetic algorithm and an ant-colony optimization (ACO) algorithm enhanced with local search. We compare the performance of these two hybrid algorithms against the solutions obtained using the greedy heuristic and another hybrid genetic algorithm proposed in literature. We find that the ACO algorithm enhanced with a minimization heuristic performs better than all other algorithms in almost all instances.

Keywords: Ant-Colony Optimization, Genetic Algorithm, Heuristic, Minimum Dominating Set.

1 Introduction

A dominating set (DS) of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that every node $v \in V$ is either a member of S or is adjacent to a member of S. A dominating set with minimum cardinality is called as the Minimum Dominating Set (MDS). This is proven to be an NP-hard problem [8]. Minimum dominating sets are extensively used in wireless networks for clustering and formation of backbone used for routing. In addition, they are also used in information retrieval, facility location and so on. Reducing the cardinality of a DS is useful in reducing the cost of locating facilities. In information retrieval, a query is matched against the dominating nodes. If the cardinality of the dominating set is reduced, the time for retrieving information is accordingly reduced.

The greedy heuristic [13] for finding a minimum dominating set (MDS) is based on that given for the set cover by Chvatal [2]. This returns a solution at most $(ln\Delta \times Opt)$ where Δ is the maximum degree of a node in the graph. It is proven to be the optimal approximate solution unless $P = NP$ [13]. In this algorithm, the nodes are all initially colored WHITE. A node which is in the dominating set is colored BLACK and all

B.K. Panigrahi et al. (Eds.): SEMCCO 2011, Part II, LNCS 7077, pp. 97–104, 2011.

dominated nodes are colored GREY. The algorithm works by selecting a non-BLACK node with the maximum white degree and including it in the dominating set. The node is then colored BLACK and all its neighbors are colored GREY. This is repeated until all nodes are dominated, i.e., there are no more WHITE nodes in the graph. A polynomial-time approximation scheme (PTAS) for the computation of MDS is presented in [9], but this is applicable only for polynomially bounded growth graphs such as unit disk graphs. A polynomially-bounded growth graph is a graph where the number of independent nodes in the r-hop neighborhood of a node are bounded by a polynomial $f(r) = O(r^c)$ for some constant $c \geq 1$. However, the PTAS algorithm is not practical for unit disk graphs (UDG) with higher degree of connectivity or large graphs. We observed that the algorithm takes a lot of time when we implemented the PTAS algorithm for Minimum Independent Dominating Set problem.

Recently, the problem of MDS has been solved using a hybrid genetic algorithm [4]. In this paper, the authors use a generational GA with a linear rank selection algorithm to select parents for the standard one-point crossover and uniform mutation to generate members of the new population. It uses three procedures to reduce the cardinality of the solution computed. One is called *Filtering* which basically checks if a node can be removed from the dominating set without affecting the domination property. Procedure *Local Search* tries to randomly add nodes to the dominating set proportional to its degree if the generated population member is not a DS; it deletes a node randomly in inverse proportion to its degree if it is a DS. In both cases, if the fitness value increases, it retains the change to the chromosome. The final procedure in the paper, *EliteInspiration* constructs the intersection of the three best DS solutions found so far and then tries to minimize the cardinality. However, in our experimentation, we found that in some cases, this algorithm does not even find a DS. We modified the algorithm such that it always returns a DS as output. We changed it as follows: after finding the intersection of the best n_{core} members of the DS population, if the resultant solution is not a DS, we keep adding nodes to it using the greedy heuristic until the member is a DS. We then, applied *Filtering* to reduce the cardinality of this solution. We found that with these changes, the solution returned by their algorithm is better than a straighforward implementation of their paper. The results presented in this paper are with these changes incorporated.

In this paper, we propose a hybrid genetic algorithm which is vastly different from that in [4] and an Ant-Colony Optimization (ACO) algorithm that uses local search to minimize the DS found by each ant in every iteration. We compared the performance of these two meta-heuristic algorithms proposed here with those in [4] and the greedy heuristic. We experimented using unit disk graphs and other general graph instances. We found our algorithms to be consistently better than the greedy heuristic and the HGA in [4]. In fact, we found that the heuristic is better than the HGA of [4]. We find that the Ant-Colony Optimization (ACO) algorithm combined with a local search heuristic performs better than all the other algorithms studied.

The rest of the paper is organized as follows: we present the steady-state genetic algorithm with the minimization heuristic in section 2 and the ant-colony optimization algorithm in section 3. The comparison of the results between the heuristic, hybrid genetic algorithm in [4] and our proposed algorithms is given in section 4. We end the paper with the conclusions in 5.

Algorithm 1. ***Hybrid Genetic Algorithm for MDS***

```
Generate Initial Population, I
F := fitness of best member of I
b := Best member of I
while gen < MAXGEN do
  if (p < p_c) then
    Select parents p_1 and p_2 using binary tournament selection
    C := crossover(p_1, p_2)
    C := mutate(C)
  else
    Generate C randomly
  end if
  if (p < p_h) then
    C := HeuristicRepair(C)
  else
    C := RandomRepair(C)
  end if
  C := Minimize(C)
  if unique(C) then
    Replace worst member of the population with C
    if f(C) < F then
      F := f(C)
      b := C
    end if
    gen := gen + 1
  end if
end while
return b
```

2 The Hybrid Genetic Algorithm

We have used a steady-state genetic algorithm [3] to solve the MDS problem. The chromosome is represented as a bit vector of size N where N is the number of nodes in the given graph. The fitness of the solution is the cardinality of the generated dominating set (DS). We start off with an initial population of 100 members. Each member is generated by randomly setting bits in the bit vector with a probability of 0.3. If the generated solution is not a DS, we repair it using the greedy heuristic with probability p_h or by adding nodes randomly. Then, we minimize the cardinality as follows: if any node of the DS and all its neighbors are covered by other nodes in the DS, it is redundant. Such a node can be removed from the DS, thus reducing cardinality without affecting domination property. We repeatedly remove redundant nodes until there are no more such nodes in the computed DS. In each iteration, we remove a redundant node randomly with probability p_r or using the policy of lowest degree redundant node with probability $(1 - p_r)$. If the solution thus generated is unique, it is added to the population. For creating a new child, binary tournament is used to select two parents for crossover. The parent with a better fitness is selected with probability p_{better}. We use the crossover method proposed

by Beasley and Chu [1] to create the child member. Let the parents be p_1 and p_2 and their respective fitness values $f(p_1)$ and $f(p_2)$. The bits in the child are inherited from parent p_1 with probability $\frac{f(p_2)}{f(p_1)+f(p_2)}$ and from parent p_2 with probability $\frac{f(p_1)}{f(p_1)+f(p_2)}$. This method ensures that bits are more often inherited from a parent with a better fitness ensuring that the fitness of the child is likely to be better. Crossover is not applied always but with probability p_c; in other instances, a new member is generated randomly to diversify the population. A simple bit flip mutation scheme is used with probability p_m. If the new child generated is not a DS, the greedy heuristic is used to add nodes until it is a DS. We, then, minimize the solution as specified in initial population generation. We replace the worst member of the previous generation with this new member. Algorithm 1 provides the pseudo-code of our approach.

3 The ACO Local Search Algorithm

First of all, we experimented with a standard ACO with no enhancements. We found the results to be much worse than even the greedy heuristic. We tried enhancing the algorithm by calculating probability based on both the pheromone value as well as the degree of a node as proposed for the minimum weighted vertex cover problem by Shyu et al [10]. In this, the higher the degree of a node, the higher the probability with which it is selected. However, we found that this was not generating a good solution either. We then proposed a local search similar to that used in the maximum clique problem in [12] as an enhancement to the standard ACO. We found this to be the most effective algorithm and this is what is described here.

Algorithm 2. ***ACO Algorithm for Computing MDS***

$F := N$
$b := \phi$
for $I := 1 \rightarrow MAX - ITER$ **do**
  for $A := 1 \rightarrow MAX - ANTS$ **do**
    $p_i := \frac{\tau_i}{\sum_j \tau_j}$
    Add node i to S with probability p_i until S is a dominating set
    $S :=$ Minimize(S)
  end for
  $D := Best(S)$
  if $f(D) < F$ **then**
    $F := f(D)$
    $b := D$
  end if
  $Update\_Pheromone(D)$
end for
return b

In our hybrid ACO algorithm, in each cycle, a total of N_{ants} perform a random walk of the graph until they discover a DS. Initially, we deposit a pheromone value of $\tau_0 = 10.0$ on each node. Each ant chooses the next candidate to include in the DS based on the probability of the node which is calculated as $p_i = \frac{\tau_i}{\sum_j \tau_j}$, where τ_i is the pheromone concentration on node i. After the random walk of an ant is completed, the DS found is minimized using the concept of redundant nodes as described in the previous section. At the end of a cycle, we use the best ant found in that cycle to reinforce the pheromone value. The pheromone concentration on the nodes of the best ant found is updated using the formula $\tau_i = \rho \times \tau_i + \frac{1}{10+f-F}$ where f is the fitness of the best ant in this cycle and F is the fitness of the best ant found so far and ρ is the pheromone persistence rate. This is similar to the formula used to update pheromone value in [11]. For all the other nodes in the graph, the pheromone is evaporated using the formula $\tau_i = \rho \times \tau_i$; if the resultant value is less than τ_{min}, the value is set to τ_{min}. We run the algorithm for a total of specified cycles. We present the results using only the local search in this paper due to lack of space.

Table 1. Cardinality (γ) of MDS and Time taken in seconds using Heuristic, Hedar, HGA and ACO-LS for UDG Instances

N	Range	Heuristic	Hedar		HGA		ACO-LS	
			γ	Time (s)	γ	Time (s)	γ	Time (s)
50	150	13.9	15.4	0.1	12.9	0.7	12.9	1.1
50	200	10.5	11.3	0.0	9.4	0.6	9.4	1.0
50	250	8	8.6	0.1	6.9	0.5	6.9	0.7
100	150	19.4	20.8	0.2	17	2.2	17	2.9
100	200	12.8	13.5	0.2	10.4	1.6	10.4	2.0
100	250	9.1	10.2	0.3	7.5	1.1	7.6	1.6
250	150	22.7	24.8	0.5	18.7	6.0	18.1	9.4
250	200	14.6	15.5	0.8	11.4	3.5	11	5.8
250	250	10.1	11.2	1.0	8	3.0	8	4.4
500	150	75.3	84.7	1.7	67.3	95.4	64.5	83.5
500	200	48.2	55.4	1.5	41.4	43.8	39.8	51.9
500	250	34.6	36.9	1.0	27.9	17.6	26.8	33.3
750	150	82.9	90.4	3.2	72.9	152.8	68.7	170.2
750	200	51.4	59	2.4	43.9	54.8	41.3	91.6
750	250	35.9	39.2	2.5	28.7	24.6	27.3	57.0
1000	150	85.9	94.2	4.4	74.8	215.1	70.3	264.3
1000	200	53	60	3.4	44.8	65.9	42.5	135
1000	250	36.7	39.8	4.0	29.8	35.5	28.2	83.9

4 Experimental Results

The experiments were done on two different types of graphs - unit disk graphs generated using the UDG topology generator [5] and Waxman Router Topologies [14] using

BRITE [7]. The data set consists of 50, 100, 250, 500, 750 and 1000 nodes. For UDG topologies, we used ranges of 150, 200 and 250 units to study the effect of different degrees of connectivity on the performance of the algorithms. Graphs with nodes 50, 100 and 250 are generated using an area of 1000×1000 units whereas those with nodes 500, 750 and 1000 are generated using an area of 2000×2000 units. In the case of Router Waxman topologies, we used random and heavy-tailed placement [6] which is considered more common for Internet topologies. We varied the degree of connectivity by using $2 \times N$, $4 \times N$ and $8 \times N$ edges for graphs, where N is the number of nodes in the graph. In all cases, the results presented are averaged over 10 instances.

The hybrid genetic algorithm that we implemented (HGA) has an initial population of 100 members and we ran it for 10,000 generations. This would mean that a total of 10,000 solutions are created as we create one new member of the population in each generation. We use the following values for the various probabilities: when generating random population members both in initial population and a new member in later generations, we use a probability of 0.3 for adding a node into the dominating set. We use the value of $p_c = 0.9$ for crossover, $p_m = 0.02$ for mutation, $p_{better} = 0.8$ to choose a better parent during binary tournament selection, $p_h = 0.2$ for using the heuristic to repair a member. We use random removal of a redundant node with probability $p_r = 0.6$. All of these values were arrived at after extensive experimentation with different values. The hybrid ACO algorithm has 20 ants and is run for 1000 iterations which is a total of 20,000 solutions. We use an initial pheromone value of 10.0 and a minimum threshold on the pheromone value of 0.08. The pheromone persistence rate is $\rho = 0.985$. The algorithm from [4] has been used with 100 generations and 100 members in the population for a total of 10,000 solutions, the same as in our hybrid GA. The rest of the parameters are as specified in their paper.

It can readily be seen that the algorithm due to Hedar et. al. [4] is performing worse than even the greedy heuristic in all the graph instances studied. As stated earlier, if not for the changes we introduced, we were getting results that were even worse. This can be explained by the fact that there is no attempt to force each member of the population to be a DS. Minimization is also not done in the best possible way. The final *EliteInspiration* process seems flawed because in standard elitism, the best n_{core} members are always retained across generations. In their method, they are doing an intersection of the best n_{core} members which is not guaranteed to even be a DS.

We observe that the hybrid ACO performs better or on par with our own HGA for most instances in terms of cardinality of the solution. This shows that it is better to use ACO-LS for MDS than the HGA. We observe that the time taken by ACO-LS and HGA are similar up to 250 nodes for UDG instances. But, as the number of nodes increases, the ACO-LS algorithm takes more time than HGA, upto twice that of HGA. However, we note that the solutions generated by ACO-LS are also twice those generated by the GA. When it comes to large Router Waxman topologies, the time taken by ACO-LS is actually smaller than that of HGA. We also observe that for large Router Waxman graphs with more degree of connectivity, the HGA performs slightly better than the ACO-LS algorithm both in terms of cardinality as well as time.

Thus, we can conclude that the ACO-LS we have presented here is the best metaheuristic approach found so far for the problem of MDS.

Table 2. Cardinality (γ) of MDS and Time taken in seconds using Heuristic, Hedar, HGA and ACO-LS for Router Waxman Instances with random and heavy-tailed (ht) placement of nodes

N	Range	Placement	Heuristic	Hedar		HGA		ACO-LS	
				γ	Time (s)	γ	Time (s)	γ	Time (s)
50	100	ht	13.5	15.1	0.1	12.1	0.6	12.1	1.0
50	100	random	12.4	14.4	0.0	11.6	0.7	11.6	1.0
50	200	ht	7.7	10.9	0.1	7	0.5	7	0.6
50	200	random	7.3	10.3	0.0	6.8	0.5	6.8	0.8
50	400	ht	4.7	6.7	0.2	4.2	0.4	4.2	0.5
50	400	random	4.1	6.6	0.1	3.8	0.3	3.8	0.4
100	200	ht	26.7	31.8	0.1	23.6	2.4	23.7	3.4
100	200	random	25.8	32.1	0.2	23.4	2.7	23.4	3.3
100	400	ht	16.5	21.5	0.2	14.8	1.8	14.5	2.4
100	400	random	16.1	22.1	0.2	14.7	1.9	14.4	2.4
100	800	ht	9.6	15	0.1	8.7	1.1	8.7	1.6
100	800	random	9.2	15.5	0.2	8.7	1.2	8.4	1.6
250	500	ht	64.7	77.1	0.6	59.4	24.6	58.5	23.3
250	500	random	67.2	78.4	0.6	60.8	25.1	59.8	23.5
250	1000	ht	42.2	56.4	0.7	38.2	15.2	37.2	17.4
250	1000	random	41.8	57.9	0.6	38.5	16.2	37.1	17.4
250	1000	ht	26.1	39.6	0.5	23.7	7.4	23.1	11.1
250	1000	random	25.3	40.2	0.6	23.2	8.2	22.4	10.9
500	1000	ht	131.4	152.7	2.0	122.2	161.7	117.3	129.4
500	1000	random	131	157.7	2.0	121.7	162.2	117.9	127.2
500	2000	ht	83.9	113.5	2.3	78.4	94.5	75.8	97.9
500	2000	random	84.1	116.7	2.3	79.5	100.7	76.5	99.6
500	4000	ht	52.3	81.1	1.7	50.3	43.9	49	61
500	4000	random	50.1	83.8	1.7	48.7	49.1	47.2	60
750	1500	ht	196.6	230.8	4.2	183.7	520.5	176.9	375
750	1500	random	195.7	241.5	4.3	185.1	527.3	178.7	372.4
750	3000	ht	125.6	169.2	4.8	119.2	302.7	119.6	288.2
750	3000	random	127.3	176.4	4.6	120.8	319.2	119	290.4
750	6000	ht	78.2	120.5	4.1	76.6	134.9	77.7	171.2
750	6000	random	77.7	128.9	4.2	76.4	140.1	75.6	169.5
1000	2000	ht	268	317	7.5	251.3	1211.4	244.8	853.7
1000	2000	random	259.8	313.5	7.4	247.6	1212.3	239.1	824.7
1000	4000	ht	168.5	231.5	7.5	161.1	720.9	163.3	635.7
1000	4000	random	168	236.9	7.7	160.7	735.6	161.3	638.2
1000	8000	ht	104.1	166.3	8.5	103	300.7	106.5	359.7
1000	8000	random	104	171.4	8.2	102.5	311.1	106.7	367.8

5 Conclusions

We have proposed a steady-state hybrid genetic algorithm (HGA) and a hybrid ACO algorithm that combines ACO with local search (ACO-LS) for the problem of Minimum Dominating Set. We compared the results obtained by these two algorithms with

the greedy heuristic which is the optimal approximate solution and a hybrid genetic algorithm proposed by Hedar et. al. [4]. We find that the ACO-LS algorithm gives the best cardinality in almost all the cases. In most cases, its running time is comparable to that of the HGA and in some cases, it is actually better. It takes more time for large UDG instances compared to HGA. However, the ACO-LS algorithm generates twice as many solutions as HGA. This implies that the ACO-LS is actually much faster than the HGA. We found that an ACO which computes probability of a node for inclusion in the random walk of an ant based on both pheromone and a heuristic such as the degree of a node performs worse than the local search algorithm presented here. Thus, we conclude that the best solution is obtained using standard ACO enhanced with a minimization heuristic.

References

1. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. European Journal of Operational Research 94(2), 392–404 (1996)
2. Chvatal, V.: A greedy heuristic for the set covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)
3. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
4. Hedar, A.R., Ismail, R.: Hybrid Genetic Algorithm for Minimum Dominating Set Problem. In: Taniar, D., Gervasi, O., Murgante, B., Pardede, E., Apduhan, B.O. (eds.) ICCSA 2010. LNCS, vol. 6019, pp. 457–467. Springer, Heidelberg (2010)
5. Mastrogiovanni, M.: The clustering simulation framework: A simple manual (2007), http://www.michele-mastrogiovanni.net/software/download/README.pdf
6. Medina, A., Lakhina, A., Matta, I., Byers, J.: Brite user manual, http://www.cs.bu.edu/brite/user_manual/node42.html
7. Medina, A., Lakhina, A., Matta, I., Byers, J.: Brite: An approach to universal topology generation. In: Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, MASCOTS 2001 (2001)
8. Garey, M.R., Johnson, D.S.: Computers and Tractability, A guide to the theory of NP-Completeness. Freeman and Company, New York (1979)
9. Nieberg, T., Hurink, J.L.: A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 296–306. Springer, Heidelberg (2006)
10. Shyu, S.J., Yin, P.Y., Lin, B.M.: An ant colony optimization algorithm for the minimum weight vertex cover problem. Annals of Operation Research 131(1-4), 283–304 (2004)
11. Singh, A., Baghel, A.S.: New metaheuristic approaches for the leaf-constrained minimum spanning tree problem. Asia-Pacific Journal of Operational Research 25(4), 575–589 (2008)
12. Solnon, C., Fenet, S.: A study of aco capabilities for solving the maximum clique problem. Journal of Heuristics 12, 155–180 (2006)
13. Wattenhoffer, R.: Distributed dominating set approximation, http://www.disco.ethz.ch/lectures/ss04/distcomp/lecture/chapter12.pdf
14. Waxman, B.: Routing of multipoint connections. IEEE Journal of Selected Areas in Communication 6(9), 1617–1622 (1988)