A new algorithm for the Maximum Independent Set Problem

Lachlan Ashcroft
Griffith University

Gold Coast, Australia
lochieashcroft@griffithuni.edu.au

***Abstract*—This paper contains a brand new algorithm for identifying maximum independent sets in a graph. A literature review was performed to give context on what the current research is in this domain and as a point of comparison for the effectiveness of the algorithm.**

***Index Terms*—independent, set, MIS, algorithm**

I. Introduction

Given an unweighted, undirected graph $G = (V, E)$ where V is set of vertices and E a set of edges, an independent set contains vertices in which no two vertices are adjacent. A maximal independent set is a set in which the inclusion of any additional vertices would introduce a connecting edge into the set. The maximum independent set (MIS) of a graph is the largest maximal independent set. The maximum size of the MIS is N, where N is the number of vertices however this is only if G is a completely disconnected graph. The MIS is a NP hard optimization problem [1].

MIS is closely related to the maximum clique problem, by definition a clique in Gc (complement of G) is an independent set in G, by this logic the maximum independent set can be found by solving the maximum clique problem in the complement of in the input graph.

There is also a relationship between MIS and the minimum vertex cover problem, however the algorithm presented in this paper focuses purely on the MIS.

The maximum independent set of a graph has various real world applications and because of this an efficient algorithm is highly desired. This paper is structured as follows, section 2 contains the literature review, section 3 the LSV2 algorithm, section 4 the results and lastly the conclusion is section 5.

Tests will be ran against a subset of the DIMACS[2] in order to benchmark the LSV2 algorithm.

A. Terminology

The degree of a vertex v shown as $d(v)$ is the number of edges (connections) of vertex v.

A neighbourhood is all vertices directly connected to an individual vertex.

The degree of the neighbourhood $dn(v)$ is the sum of the degrees of all vertices in the neighbourhood.

Adjacency matrix is a matrix of dimensions (N, N) where N is the number of vertices, a value of 1 indicates two vertices are adjacent.

Adjacency list/set contains all the vertices which are connected to an individual vertex.

II. Literature Review

A literature review was performed in order to understand the current state of the research regarding the MIS, used as inspiration for the development of a new algorithm, and lastly these algorithms can be used to assess the effectiveness of the developed algorithm.

As mentioned in the introduction the MIS is related to both the maximum clique and minimum vertex cover problems. During the research it was found that many approaches to solve the MIS is to simply translate the problem into one of these other problems, find a solution and convert back. This is a valid solution for the MIS and should still be included in the literature review to some degree, even though the developed algorithm explicitly solves the MIS.

[3] Michael Luby converted Monte Carlo style algorithms into deterministic algorithms with the same parallel running time in 1986. The expected running time of the entire algorithm is $O((logn)^2)$ using $O(m)$ processors. This alone shows that parallel randomized algorithms may be an efficient approach for solving the MIS. Another parallelised algorithm proposed at the University of California at Berkley [4] has an expected running time of $O((logn)^4)$ using $O(n^3/(logn)^3)$ processors, the randomized version of the algorithm is $O((logn)^3)$ with $O(n^2)$ processors.

Both of these algorithms are of order $O(logn)$ but to achieve this run time a large number of processors are required, which not feasible for a significant sized of n which is the number of vertices in the graph.

In 1995 a greedy randomized adaptive search algorithm was proposed [5]. In the first phase it iteratively builds a set with a random greedy approach, the heuristic values for the vertices are then updated every iteration. This algorithm can also be ran in parallel which increases the effectiveness. The testing went up to 8 processors and the results are very promising, especially considering if this were to be ran on modern hardware.

An algorithm states it has a run time of $1.1966^n n^(O(1))$ and $1.1893^n n^(O(1))$ for graphs with a maximum degree of 6 and $1.1970^n n^(O(1))$ for 7 [6]. It does this by making a careful analysis on the structure of these bounded degree

graphs. These expected run times are very good compared to others however they are very limited in their application. An extension could be to split a graph into multiple sub graphs in which each sub graph would of equal vertex degrees, use this sort of algorithm to the find the MIS, and then combine sets to find the MIS of the entire graph.

As stated before the MIS can be solved by finding the maximum clique in the input graphs complement, in 2013 a new algorithm was developed [7]. A single threaded algorithm was developed which is a branch and bound derivative that happend to out perform existing algorithms. This algorithm was then parallelised distributing the search tree between multiple CPU cores, using 12 CPU cores that algorithm provided up to 2 orders of magnitude faster execution on large benchmarks.

An algorithm designed with a run time complexity of $O(n^4)$ has been developed [8]. The algorithm is greedy based and is recursive. It picks the vertex with the highest degree, all vertices not connected are put into a set, if collisions are present they the vertex of the highest degree is removed, and this process is repeated. A graph is included showing the relationship between the time taken to find the MIS and the graph density with 1000 vertices, there is a direct correlation between a lower run time and high density graphs.

Simulated annealing is an iterative improvement algorithm which incorporates randomness, it has been used to solve many problems, the MIS included. Using simulated annealing near optimal solutions can be found with a high probability of finding the optimal solution - the MIS of the graph [9]. As shown in the paper the MIS of the graph is not always found but is generally very close to the optimum, this same property is shown when compared to other algorithms in the result section.

As mentioned in the introduction a solution to the MIS is the maximum clique in the complement of the input graph. In 2006 an algorithm was published called DLS-MC which is a stochastic local search algorithm that alternates between iterative improvement and plateau search. [10] The algorithm has some very interesting concepts and in the results section is shown to be very competitive with the existing algorithms.

A genetic algorithm is an iterative improvement method, it has parent solutions and then generates offspring solutions. The paper outlines a heuristic crossover method which ensures that the offspring are valid independent sets [11], this is where the success of the algorithm is owed. Offspring are generated by mutating solutions and because of this the resulting solutions are often invalid, this algorithm ensures all offspring and therefor future parents are valid solutions. The result sections shows that this algorithm is able to reach the optimum solution with a high probability and if not is at least extremely close.

III. The LSV2 Algorithm

The LochieSolverV2 (LSV2) algorithm takes inspiration from the Fibonacci heap datastructure's pop min behaviour in which the trees are merged together. The algorithm creates and merges multiple independent sets as well as collapsing sets. As seen in the results section this approach on average finds an IS that is close to the global maxima and also is very quick. The algorithm is stochastic and therefor a guarantee cannot be given for finding the MIS, to increase the probability this algorithm is run multiple times. Each IS stores a map that contains the set's adjacent vertices as well as how many vertices in the set are adjacent to it. This map is crucial to the algorithms performance, it determines if a vertex can be added to the set and maintain the IS characteristics efficiently and is updated on every insert and removal. LSV2 has four stages, the initial pass, initial merge, the main loop and collapsing. To begin there are K empty independent sets, every vertex in the graph is attempted to be added to these sequentially, the vertices which were unable to be included are added back to the pool of available vertices. Next comes the initial merge which attempts to combine all of the K IS together. The main loop attempts to insert a random vertex into the current sets if unable to a new set is created and after L insertions all the sets are merged. This process continues until there is no available vertices left, in which case a random set is collapsed and it's vertices are made available and the process continues for a specified amount of times.

Algorithm 1 LSV2

Output: MIS size

while $count \leq limit$ **do**
  $nodesQueue = Queue(Randomize(nodes))$
  $sets = set[k]$
  Perform initial pass
  Perform initial merge
  $shouldMerge = 0$
  $collapseCount = 0$
  while $nodesQueue$ is not empty **do**
    $node = nodesQueue.pop()$
    if $sets.insert(node)failed$ **then**
      $sets.create(node)$
    end if
    $shouldMerge$ += 1
    if $shouldMerge$ == 5 **then**
      $sets.merge()$
      $shouldMerge$ == 1
    end if
    if $nodesQueue$ is empty and $collapseCount \leq 4$ **then**
      $sets.collapse()$
      $collapseCount$ += 1
    end if
  end while
  $count$ += 1
end while
return $largest$

Algorithm 2 Merge

```
for node in B do
    if node not in A then
        if A.insert(node) then
            update A Adjacency Map
            update B Adjacency Map
            B.remove(node)
        end if
    end if
end for
```

Algorithm 3 Insert

```
for set in sets do
    if set.insert(node) then
        update Adjacency Map
        return True
    end if
end for
return False
```

IV. Results

A subset of the DIMACS benchmarks were used to test the effectiveness of LSV2. In addition a target algorithm, greedy algorithms using the degree of a vertex (Greedy D) and the neighbourhood degree (Greedy ND) for the heuristic and a random approach are displayed. The LSV2 configuration was 1000 loops, 4 initial empty sets and to merge all sets after every 5 insertions. NOTE: The collapsing of the sets was not incorporated for the below results.

Graph	Target	LSV2	Greedy D	Greedy ND	Random
brock800_1	10	8	8	8	9.3
brock800_2	10	8	8	8	9.3
brock800_3	11	8.1	7	7	9.4
brock800_4	10	8.2	7	7	9.4
C2000.5	17	12.9	12	10	14
C2000.9	6	5.2	5	5	5.8
C4000.5	18	14	13	13	15
keller6	63	61.4	14	20	63
MANN_a5	3	3	2	2	3
san1000	67	54.7	67	67	64.2

TABLE I: Average size of the MIS

From the table above it is clear that LSV2 finds larger maximal IS compared to the Greedy Algorithms, typically under that of the target however. MANN_a5 is interesting as both Greedy Algorithms are unable to reach the target but LSV2 is. Keller6 is similar however the delta is much larger for the Greedy Algorithms. The Greedy Algorithms are deterministic, the LSV2 results were averaged from a series of runs which is why there are decimals, if only considering the highest value than LSV2's results would be higher.

The CPU times is where LSV2 shines there is not that much of an overhead compared to the Greedy Algorithms

Graph	Target	LSV2	Greedy D	Greedy ND	Random
brock800_1	0.02	0.013	0.007	0.007	1.103
brock800_2	0.02	0.013	0.016	0.17	1.243
brock800_3	0.43	0.013	0.011	0.011	1.377
brock800_4	0.01	0.013	0.006	0.006	1.079
C2000.5	561.02	0.069	0.064	0.064	5.507
C2000.9	0.02	0.137	0.068	0.068	2.017
C4000.5	285.17	0.273	0.356	0.359	17.925
keller6	0.08	0.104	0.913	0.913	1.189
MANN_a5	$\leq\epsilon$	$\leq \epsilon$	0.005	0.005	$\leq \epsilon$
san1000	$\leq\epsilon$	0.008	0.011	0.011	5.3474

TABLE II: Average CPU Time

but in return larger IS are highly likely. LSV2 generally outperforms the Target algorithm in terms of CPU time and this is extremely evident on the graph C2000.5. These CPU times can be attributed to the hardware that the algorithms were ran on and in terms of LSV2 the efficient datastructures used mostly from the STL which had O(1) operations. All in all LSV2 outperforms Greedy generally and should be used instead, the Target algorithm has higher accuracy but has a longer running time, LSV2 is a good compromise.

V. Improvements and Experimentation

There are a number of potential experiments that could be performed which may improve LSV2. These include the number of starting sets and how these are populated - currently randomly but there is no reason why it could not be greedy. Secondly modifying the insert / merge behaviour from instead of performing sequentially it could be performed randomly. The merging of all the sets could be initiated randomly instead of at fixed intervals and lastly the collapsing of the set, this was not functional for the above results. High density graphs would be an interesting aspect to explore as the results from LSV2 maybe continue to outperform that of a Greedy Algorithm.

References

[1] Garey. M. R, Johnson. D. S: Computers and Intractability: A Guide to the theory NP – completeness. San Francisco: Freeman (1979).

[2] DIMACS clique benchmarks. Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, Rutgers Univ., Piscataway. NJ. (1993).

[3] Luby. M: A simple parallel algorithm for the maximal independent set problem: - SIAM journal on computing (1986)

[4] Richard M. Karp, Avi Wigderson: A Fast Parallel Algorithm for the Maximal Independent Set Problem: - Journal of the Association for Computing Machinery. Vol 32, No. 4 1985

[5] Thomas A. Feo, Mauricio G. C. Resende, Stuart H. Smith: A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set: - Operations Research Vol. 42, No. 5 1994

[6] Mingyu Xiao, Hiroshi Nagamochi: Exact Algorithms for Maximum Independent Set: - Information and Computation, 08/2017, Volume 255

[7] Matjaž Depolli†, Janez Konc, Kati Rozman, Roman Trobec, and Dušanka Janežič: Exact Parallel Maximum Clique Algorithm for General and Protein Graphs - J. Chem. Inf. Model., 2013, 53 (9), pp 2217–2228

[8] Ahmad Abdel-Aziz Sharieh, Mohammed H. Mahafzah, Ayman Al Dahamsheh: An Algorithm for Finding Maximum Independent Set in a Graph: - European Journal of Scientific Research ISSN 1450-216X Vol.23 No.4 (2008), pp.586-596

[9] Xu X., Ma J., .Wang H. (2006) An Improved Simulated Annealing Algorithm for the Maximum Independent Set Problem. In: Huang DS., Li K., Irwin G.W. (eds) Intelligent Computing. ICIC 2006. Lecture Notes in Computer Science, vol 4113. Springer, Berlin, Heidelberg

[10] Wayne Pullan, Holger H. Hoos: - Dynamic Local Search for the Maximum Clique Problem: - Journal of Artificial Intelligence Research 25 (2006) 159-185

[11] Mehrabi S., Mehrabi A. and D. Mehrabi A. (2009). A NEW HYBRID GENETIC ALGORITHM FOR MAXIMUM INDEPENDENT SET PROBLEM . In Proceedings of the 4th International Conference on Software and Data Technologies - Volume 2: ICSOFT, ISBN 978-989-674-010-8, pages 314-317. DOI: 10.5220/0002253403140317