copyright GBL rev t 5/20

**MIS Global Depth-First-Search Back-Tracking Design  
*[Use our Disciplined PD/AD Algorithm Top-Down Design Technique!]***

The required gs-dfs/bt (global search depth-ﬁrst-search with backtracking) by definition searches the entire MIS search space implicitly or explicitly. The dfs process searches iteratively or recursively by traversing from one partial solution state space candidate to another until maximal solutions are found. It executes backtracking to find **all** maximal/maximum (or minimal/minimum) solutions.

1. **General discussion of graph MIS problem domain:**

*“The maximal (maximum) independent set problem –* [*MIS*](https://en.wikipedia.org/wiki/Maximal_independent_set)*”*

*The MIS problem is the following: given a graph G = (V;E) and an independent set in G*

*of maximal cardinality. In the weighted case, each node v in V has an associated non-negative weight w(v) and the goal is to find maximal weight independent sets S. {v is in S and the neighborhood of v intersecting with S is not empty. This problem is NP-complete and thus it is natural to look for MIS/Clique approximation algorithms.*

- MIS informal description: (Christoﬁdes, p30-31 [1]) “internally stable set,”   
 (cliques of complementary graph) – a NP-Complete problem (NP-Hard)

- MIS formal model with n is the number of vertices: (Christoﬁdes, p31) ;   
- solution space: O(3n/3) (Christoﬁdes, p53; Reingold, Nievergelt and Deo, p353)   
- graph search space: O(2n), power set of possible vertex partial combinations

- applications: NPC, constrainted resource task scheduling, communica­tions, clique, vertex cover, ... *(note that one should be able to map in order of (p(n)) from any of these problems to and from the MIS formulation - NPC)*

**Proof that MIS is NP-Complete:**

First, we need to show that the MIS problem is in **NP**. We can show that a set |S| = k is may be an independent set or not in polynomial time.Second,the data structure of the 3-SAT problem (NP-Complete) can be reduced in polynomial time to the MIS problem, Thus, the 3-SAT can be solved using the MIS problem and therefore MIS is **NP-Complete** (GJ)**.**

**MIS Algorithm Domain (gs-dfs/bt) Development**

- selected algorithm: gs-bfs/bt search (Christoﬁdes, p33-35); set operations needed

The standard problem domain requirements analysis and generation of the high level design can be approached in the general pattern as indicated in the previous lecture notes. Mapping the global gs-dfs/bt algorithm to the design of a MIS search algorithm results in a tree search control structure (Christoﬁdes, pp33-35). The algorithmic design philosophy of Talbi [4] and Michaelewicz and Fogel [5] are also followed throughout this development. What are “good” heuristics?

* *Comment:* we design and analyze the MIS problem and study in detail the graph search (tree search) algorithm (Christoﬁdes) using the gs-dfs/bt search algorithm template. Individually you can study other algorithm implementation and applications for solving the MIS problem based upon other selected graph data structures (see references). *Observe that ADT(D,F,A) needed for sets in this algorithmic design*. For detailed MIS discussions try MIS on wikipedia. Also consider complementary [Clique problem](https://en.wikipedia.org/wiki/Clique_problem) description! Or a [Clique graph theory](https://en.wikipedia.org/wiki/Clique_(graph_theory)) description.

**MIS Problem Domain & Algorithm Domain Formal Development**

1. Problem Domain Requirements Specification form: (Christofides, p31)

- domains, D

input Di - Graph G(X,Γ), X:vertices. Γ: vertex link set (adjacency)

output Do - Maximal (Maximum) Independent sets per Christofides

- I(x); input conditions on input domain satisﬁed; x in X, link in Γ

- O(x,z); output conditions on output/input domain satisﬁed; i.e.,

a feasible/optimal solution with respect to the input domain   
-- S intersection Γ(S) = φ; independent set S in X (PD eqn 3.1 Christofides)  
-- H intersection Γ(H) not = φ for all H set in S; maximal independent set z,   
 (PD eqn 3.2 Christofides)  
-- max |S| of maximal independent sets is maximum independent set(s)   
 (PD eqn 3.3 Christofides)

1. Problem Domain/Algorithm Domain Integration Specification  
     
   *”Integrate MIS problem domain with gs-dfs/bt algorithm domain”*

* **Basic search constructs** for gs-dfs/bt (a tree search by construction!)

* *next-state-generator* (Di) − > x in X; I(x)
* *selection* (Di) − > x; x in X (usually from an ordered/sorted set based   
   explicitly/implicitly MIS criteria-desire terminal nodes to be MIS

(call the set Qk = X for each level/search-stage k of search tree!)

* *feasibility* (x, Dp) − > boolean (if true union (x, S)), S intersection Γ(S) = φ; independent set S in X
* *solution* O(x,z) “maximal “; (Dp) − > boolean; z = Dp, i.e., can no longer   
   augment S with an x in X;
* *objective (*Dp*) ->* Do *“ordered set/*[*well founded set*](https://en.wikipedia.org/wiki/Well-founded_relation) *of MI sets is regt’d”*
* **Delay Termination** from gs-dfs/bt
* *Find all* maximal independent solutions within tbd designed *loop*
* *Generate* via gs-dfs/bt all MIS solutions without duplication!
* imports: ADT( set, set-of-sets):Di Dp Do; Boolean; integer

- *Comment:*   
A) need a speciﬁc function/algorithm (unknown) that maps input domain to output domain

B) can explicitly deﬁne axioms, A; i.e., deﬁne input/output general requirements logically for testing algorithm (including exceptions)

C) consider “better” ordering in the set of candidates based upon the # of vertex connections, …. (vertex ordering)

3. Algorithm Domain Design Speciﬁcation Refinement

* *Possibly Sort a priori nodes/vertices in* the set of candidates *Qk based upon # of connections to other nodes? How to handle (store, process) PDs with very large number of nodes? Distributed or parallel computation?*
* *Creative data structure augmentation* of the set of candidates Qk  into Q+k and Q-k in gs-dfs/bt that provides for *generating sets without duplication*; a search tree vs. search graph (Christofides, pp 33-34; “Bron-Kerbosch Algorithm”)
* Observe that k is the stage index (level in gs-dfs/bt search tree): S = Sk is defined as the independent set of PD graph vertices at stage k in the tree search; Sk is a partial MIS solution.
* ***Next-State Generator and Selection:*** (CREATIVE!)
  + Q+k : set of vertices not selected previously at state (level) k or higher in search tree to augment Sk : updated with forward search *selecting* xik from Q+k ; Q+k+1 = Q+k – Γ(xik)- {xik}, *(AD eqn 3.6 from PD eqn 3.1*- Christofides)

***Additional Heuristic1: add discussion of “vertex ordering” with some math/symbols***

* The vertex ordering is based on the degeneracy ordering D, which can be found in linear time by repeatedly selecting the vertex with the smallest degree [1]. Q+k+1 will now be generated as follows: Let {d} be the set of vertices from Q+k that have the smallest degree (their degree will all be equal). Then Q+k+1 = (Q+k – Γ(xik)- {xik}) ∩ {d}

[1]<https://en.wikipedia.org/wiki/Bron%E2%80%93Kerbosch_algorithm#With_vertex_ordering>

***Additional Heuristic2: add discussion of “pivoting” with some math/symbols***

* Pivoting as described in Christofides chapter 3 is based on the use of equation 3.9 to try and force backtracking as early as possible and avoid unnecessary branching in the search tree. Equation 3.9 is

∆(x) = | Γ(x) ∩ Q+k | which in English states that delta X is equal to the cardinality of the intersection between the neighbors of x and Q+k (or more simply, the number of x’s neighbors which are in Q+k ). By choosing the next xik as described on page 35 of Christofides (shown later in this document) it guarantees that the value ∆(x) will become smaller and bring us closer to equation 3.8 becoming true thus beginning the backtracking step.

* ***Feasibility*** (CREATIVE!)
  + Q-k set of vertices which have been selected previously at state k – 1 or higher in search tree to augment Sk; removal of Γ(xik ) and xik added when backtracking from Q−k  (Q-k+1 = Q-k – Γ(xik) ) where Γ(xik) = vertices adjacent to xik ). *This is a very creative selection of a “reﬁned” data structure. (updated with equation 3.5 - Christofides with backward search when deselecting xik from Q+k ; addition of xik to Q-k and minus Γ(xik ).* *WHY?!* *Generates sets without duplication!*
* ***Solution:*** if Q+k = Q-k = : a set Sk is a MIS solution if it cannot be augmented further, and since sets are generated without duplication, Sk is a MIS solution if and only if Q+k = Q-k =   *“again a very creative insight from AD to PD!” – (indirectly from PD eqn 3.2; see Christofides for more discussion details)*

[*One can add new gs-dfs/bt heuristics:* data structures, search constructs and algorithmic operational process refinements (improved program design in a program name: MIS gs-dfs/bt program?)]

*(Improved search = fewer search nodes/branches?)* [*NFL Thm*](https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf) *impact!?****Additional Heuristic2: add discussion of “pivoting” with math/symbols and algorithmic process***

* Building upon the discussion above we will now show how to use equation 3.9 to find the next xik . First iterate through Q-kto find a vertex y with the smallest ∆(y). Next, let {p} be the set formed by Γ(y) ∩ Q+k . Pick xik to be any element of that set and it will guarantee that that at every backtracking step equation 3.8 will become closer to true.
* Continuing program development by instantiating more gs-dfs/bt search elements for backtracking loop:
* *initialize* sets Sk = Q-k = , Q+k = X, k = 0.
* *loop*
* *next-state-generator* (Di) − > xik in Q+k ; I(x)
* *selection* (Di) − > xik; xik in Q+k (usually from an ordered\* set based explicitly/implicitly MIS criteria-desire terminal nodes to be MIS)

update Q+k+1 = Q+k – Γ(xik) - xik, ; Γ(xik) = vertices adjacent to xik

* *feasibility* Q-k+1 = Q-k – Γ(xik); (xik in Dp) − > boolean (if true union (xik, Sk)), Sk   
   intersection Γ(Sk) = φ; independent set Sk in Dp with Qk construction, only   
   feasible sets are generated!
* *solution* O(xik,z); (xik in Dp) − > boolean; z = Dp, i.e., can no longer   
   augment Sk with an xik in X; Q+k = Q-k = 
* *ﬁnd all* maximal independent solutions within *loop* by *backtracking*

\*Could be lexigraphical (Christofides); input/output degrees sorted, …

* imports: integer/real/character, BOOLEAN, ADT (Set, Set-of-Sets), ...

(list of other design speciﬁcations, ADTs-algebraic specs

* data dictionary (dfs local decision creativity!)

1. Algorithm Domain Design Continuing Refinement

* Design Speciﬁcation Name: (list of parameter speciﬁcations) domains: Di,Do“MIS gs-dfs/bt Program”  
   *[Christofides algorithm does not use a priori sorting or consider # of nodes]*
* *Creative* logic data structures Q+k and Q-k regarding backtracking condition
* ***Creative*** *early backtracking* If x in Q-k so that Γ(x)  Q+k = ; i.e., if for some x in Q-k exists for which Γ(x)  Q+k = , then regardless of which x vertex is taken from Q+k to augment Sk forward, x can never be removed from Q-k (*creative equation 3.8!*)
* gs-dfs search constructs and algorithmic operational process *(continue refinement)*
* *imports:* integer/real/character, BOOLEAN, ADT (SET, SET-OF-SETS, graph), ...

(list of other design speciﬁcations, ADTs-algebraic specs,

data dictionary (dfs local decision creativity!)

* *initialize* sets Sk = Q-k = , Q+k = X, k = 0.

*loop*

* *next-state-generator* (Di) − > xik in Q+k ; I(x)

***Additional Heuristic1: add discussion of “vertex ordering” with complete math/symbols and process***

* The next state-generator will be augmented as described above in that now the vertices in Q+k will be based upon those that have the smallest degree.
* *selection* (Di) − > xik; xik in Q+k (usually from an ordered set based explicitly/implicitly MIS criteria-desire terminal nodes to be MIS)

update Q+k = Q+k - Γ(xik) - xik, ;Γ(xik) = vertices adjacent to xik

* *feasibility* Q-k = Q-k – Γ(xik); (xik in Dp) − > boolean (if true union (xik, Sk)), Sk   
   intersection Γ(Sk) = φ; independent set Sk in Dp with Qk construction, only   
   feasible sets are generated!   
   If xik in Q-k so that Γ(xik)  Q+k = , then backtrack
* *solution* O(xik,z); (xik in Dp) − > boolean; z = Dp, i.e., can no longer   
   augment Sk with an xik in X; Q+k = Q-k = 
* *backtrack to loop* until all possible combinations (states) are check implicitly or explicitly; backtrack to previous level k-1 search tree level   
   and loop; if all PD vertices have been used at the k = 0 level; i.e.,   
   Q+k =  for k = 0, then STOP.
* *axioms*: tbd (list of axioms relating parameters, types, imports, and operations) for all x in Di, if I(x) then there exists a function Fn(x) = z with z in Do that satisﬁes O(x,z); desired to find a specific function(x)/operational mapping.
* ***Comments:***   
  ***a***. Could put search construct flow in a table form for ease of understanding.

***b***. Observe that at this design level, the details of the functional implementa­tion are yet to be deﬁned; i.e., one must reﬁne the AD into a gs-dfs/bt low level design for mapping to a given computer language.   
  
***c.*** Also, the maximum independent set(s) of vertices may be required which would need a max set operation.

1. Functional Algorithm Speciﬁcation for MIS gs-dfs/bt:

*”Top-Down Design ﬂow into the Bron and Kerbosch Algorithm* *(Christoﬁdes)   
MIS dfs-bt search graph algorithm, gs-dfs/bt (page 35) [1,2]”*

“Functional MIS gs-dfs/bt Algorithm Psuedo code found in Christofides;  
 **Note:** algorithmic step-by-step math/symbolic notation! ”  
\*\*\*\*\*  
Name: **MIS gs-dfs/bt Algorithm** *(Christofides, Bron and Kerbosch)*  
*Declaration and Initialization*Step 0 *declaration: i*nteger/real/character, Boolean, ADT (set, set-of-sets), …

***Additional Heuristic1: add “vertex ordering” with math/symbols and complete algorithmic process in proper position.***

**Step 1** *Initiation*: : Let {d} be the set of vertices from X that have the smallest degree. Set Sk = Q-k = , Q+k = {d}, k = 0.  
*Forward Step* (dfs loop)   
**Step 2** *Selection: If* Q-k ≠ choose the vertex y with the smallest ∆(y) (equation 3.9 defined above). *Choose a vertex* xik from the set formed by Γ(y) ∩ Q+k *. Else,* choose a vertex xik in Q+k

Sk+1 = Sk  xik, k = k + 1

Update Q+k+1 = (Q+k - Γ(xik) - xik) ∩ {d}, where {d} is the set of vertices in Q+k with the smallest degree and where Γ(xik) = vertices adjacent to xik

*Test*  
**Step 3** *Feasibility:* Q-k+1 = Q-k – Γ(xik). If xik in Q-k so that Γ(xik) Q+k = , go to Step 5,   
 else go to step 4  
**Step 4** *Solution:* If (Q+k = Q-k = ) then PRINT MIS Sk, go to Step 5, If Q+k =  and   
 Q-k not =  go to Step 5, else go to Step 2.  
*Backtrack* **Step 5** *Loop Backtrack:* Set *k = k - 1*. Sk = Sk+1 - xik, Q+k = Q+k - xik, Q-k = Q-k + xik,   
 if k = 0 and Q+k = , STOP, else go to step 3 (dfs loop).

***Additional Heuristic2: add “pivoting” with math/symbols and complete algorithmic process in proper position.***

\*\*\*\*\*  
6. Mapping to chosen computer language

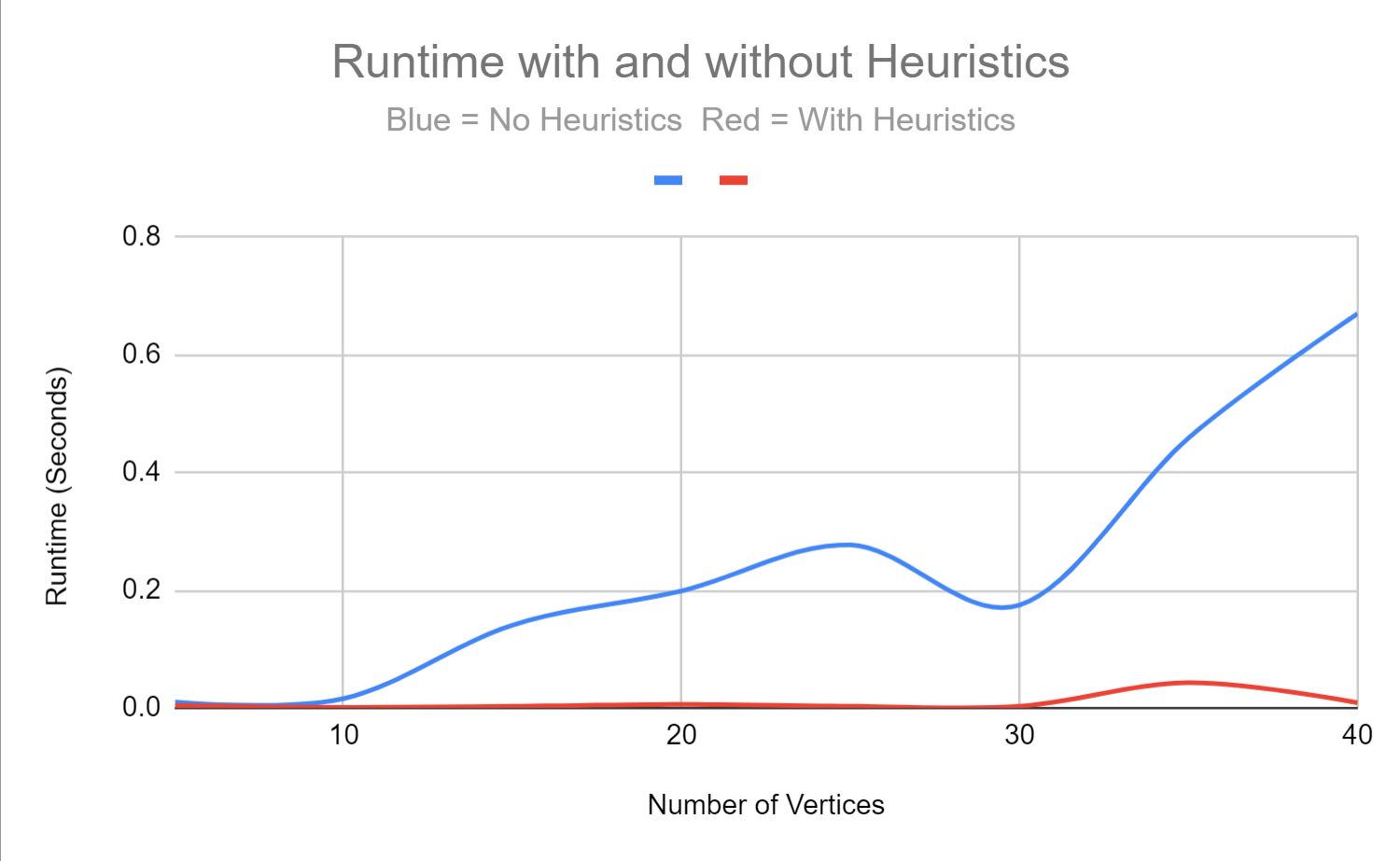
Carry standard search elements into computer language as comments. One should be able to follow the design from problem domain and algorithm domain integration explicitly to implementation.

1. Test and Evaluation Report of Software Execution

The following tests are being run on an Intel Core i7 processors @ 2.9GHz with 16GB of RAM. The programming language is C++17.

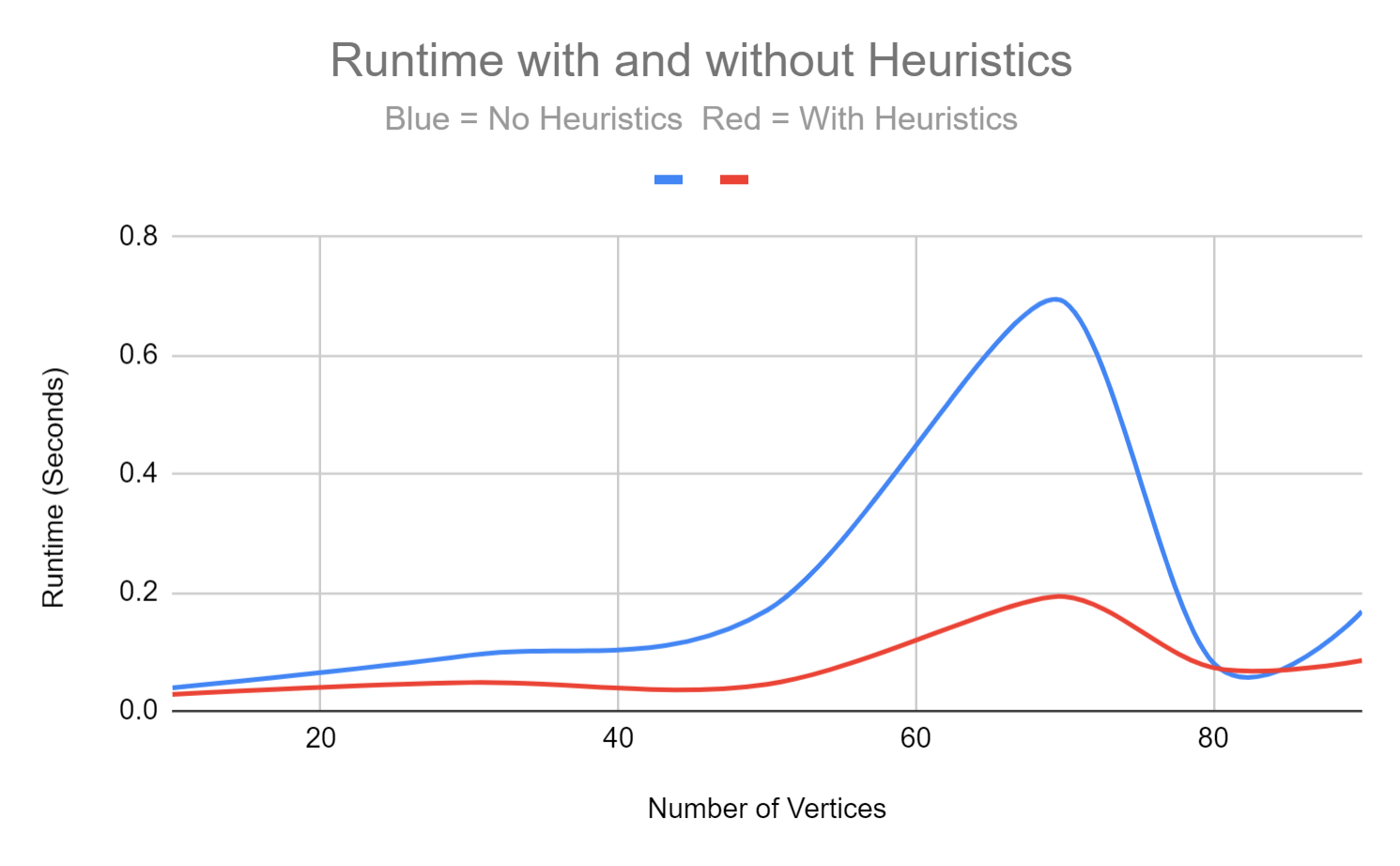
Two experiments will be performed. First randomly generated graphs with 5-40 vertices will be tested using the original version and the heuristics version to compare execution time. The main objective of the first experiment though is to verify correctness against the version with no heuristics which is known to be correct. The second experiment will use complete graphs of size between 10-100 vertices. The similar structure will give better insight into the difference in execution times. The main objective of this experiment will be to compare execution times.

**Experiment 1 Results**

**

The Heuristic version of the software always produces the correct (optimal) answer when verified against the known correct original version. The execution time is clearly significantly reduced when using the heuristic version. Larger graphs will be tested to look for further trends in execution time.

**Experiment 2 Results**

**

Using only complete graphs the heuristic version still outperforms the original by a significant factor. An interesting phenomenon occurred where complete graphs of size 70 vertices took more execution time than larger graphs. It appears that for complete graphs that size will produce the peak in computational complexity.

***Some References:*** *(also consider papers in MIS\_Clique paper directory)*

[1] Nicos Christofides. *Graph theory: An algorithmic approach (Computer science and applied mathematics)*. Academic Press, Inc., Orlando, FL, USA, 1975.

[2] Alessio Conte, [*Review of the Bron-Kerbosch algorithm and variations*](http://www.dcs.gla.ac.uk/~pat/jchoco/clique/enumeration/report.pdf), Univ of Glasgow,   
School of Computing Science, May, 2013

[3] Edward Reingold, Jurg Nieverelt, and Narsing Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice Hall, 1977.

[4] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. *The worst-case time complexity for generating all maximal cliques and computational experiments.* **Theor. Comput. Sci.,** 363:28–42, October 2006.

[5] El-Ghazali Talbi. *Metaheurisics From Design to Implementation*. Wiley and Sons, 2009

[6] Michaelewicz and Fogel, *How to Solve it: Modern Heuristics*, 2ed, Springer, ‘04

[7] Wikipedia, MIS and Clique

[8] Robson, [*Algorithms for Maximum Independent Sets*](https://www.cs.umd.edu/~gasarch/TOPICS/sat/robson.pdf), Journal of Algorithms 7, pp 425-440, 1986