Mobile and Pervasive Computing

Kasidakis Theodoros (2258)

Dominating Set Algorithms for Wireless Sensor Networks Survivability

Course Instructor : Dimitrios Katsaros

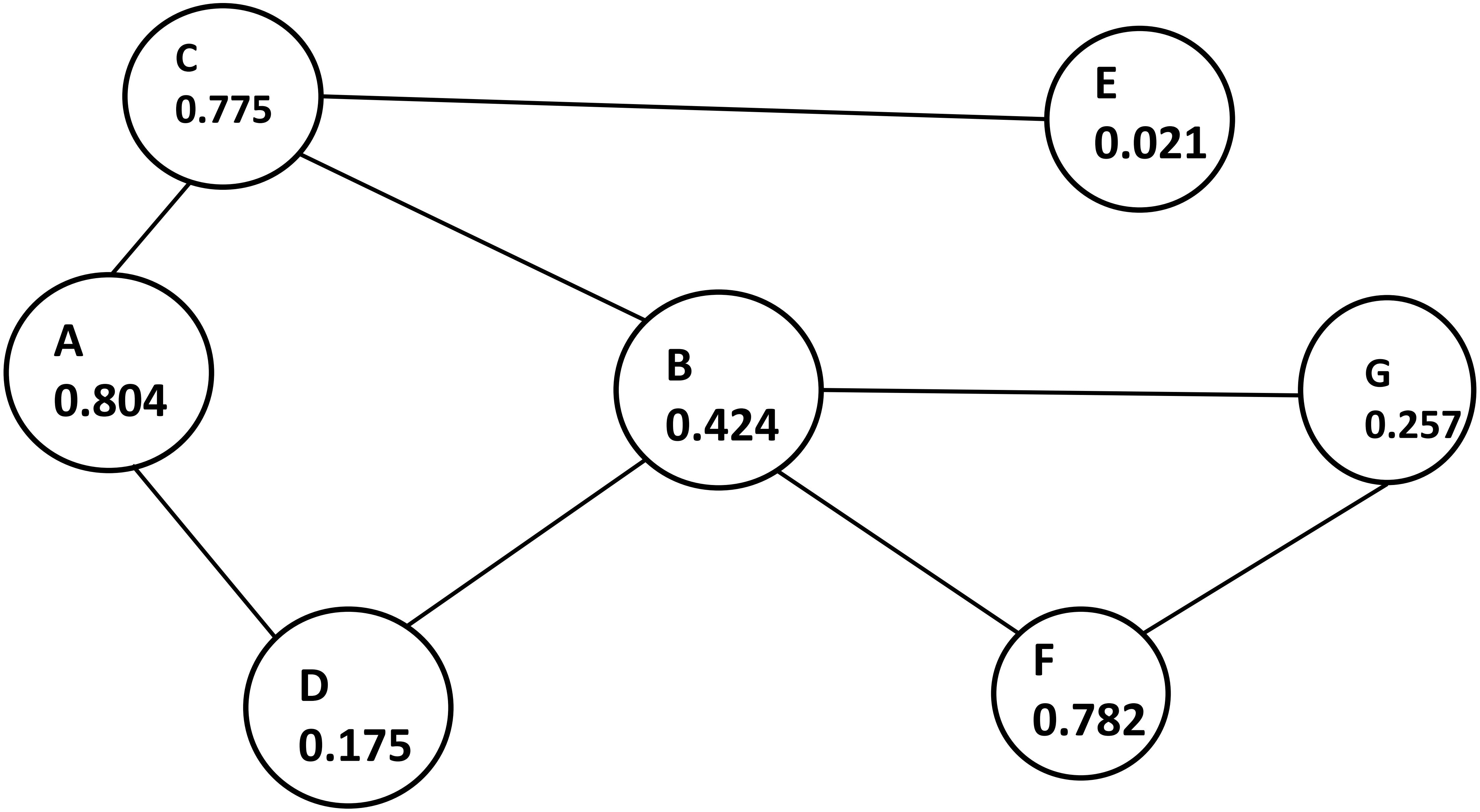
Introduction

Limited energy of the sensors is one of the key issues towards realizing a reliable wireless sensor network (WSN), which can survive under the emerging WSN applications. A promising method for conserving the energy of these sensors can be implemented by applying a sleep-wake scheduling while distributing the data gathering and sensing tasks to a dominating set of awake sensors while the other nodes are in a sleep mode. Producing the maximum possible number of such disjoint dominating sets, called the domatic partition problem in unit disk graphs, can further prolong the network lifetime. This problem becomes challenging when the initial energy of the nodes varies from one to another.

System model and problem statement

We model the wireless sensor network using an undirected graph G = (V, E) where the vertex set V denotes the set of all nodes in the network and the edge set E denotes the communication link (v1, v2) ∈ E between node v1 and v2 where v1, v2 ∈ V and they are within the communication range of each other. In our system, all the nodes have the same communication range. Any subset $S \subseteq V$ is considered to be a dominating set if every vertex $v \in V \setminus S$ has at least a neighbor in S. The research question that is to improve the total lifetime of a WSN having nodes with varying initial energy. The lifetime of a dominating set is the lifetime of the node having the minimum energy in that set divided by the rate at which the energy of the node is depleted. In our system, we assume that the rate of energy depletion is constant for every sleeping node and it is also constant for every awake node. The total lifetime of the network is then calculated by summing the lifetime of each of the disjoint dominating set. Consider the WSN consisting of the nodes A, B,C, D, E, F and G in Figure 1. The figure shows the initial energy of each of the nodes and also their neighbors. Two disjoint dominating sets [A, B, E] and [C, D, F] can be created from this WSN. In the set [A, B, E], the node E has the minimum initial energy (0.021). In the other set, the node D has the minimum initial energy (0.175). So the lifetime of the first dominating set is 0.021/0.05 = 0.42 and the second dominating set is 0.175/0.05 = 3.50. So, the total lifetime of the network is 3.92. However, this lifetime can be further improved if the disjoint sets were [B, D, E] and [A,C, F]. In this case, the total lifetime would be (0.021/0.05) + (0.775/0.05) = 15.92.

Figure 1



Algorithms

The proposed algorithm use the minimum dominating set algorithm proposed by Islam,Akl,Meijer for its initial feasible solution The minimum dominating set algorithm is a centralized algorithm which finds multiple dominating subsets S1, S2, S3, ..., Sm for a given set of nodes V. Initially, the algorithm finds the node vi from V such that vi has the largest number of neighbors among all nodes in V. Now the algorithm repeats the same process by disregarding the node vi from the previous step and all of its neighbors. In this way, the algorithm finds the minimum dominating set. The algorithm repeats the process again while discarding all the nodes that were selected in the previous steps. The algorithm continues to find additional dominating sets and stops when no such set can be found anymore. Now this list of sets is used by our proposed algorithms which are discussed below.

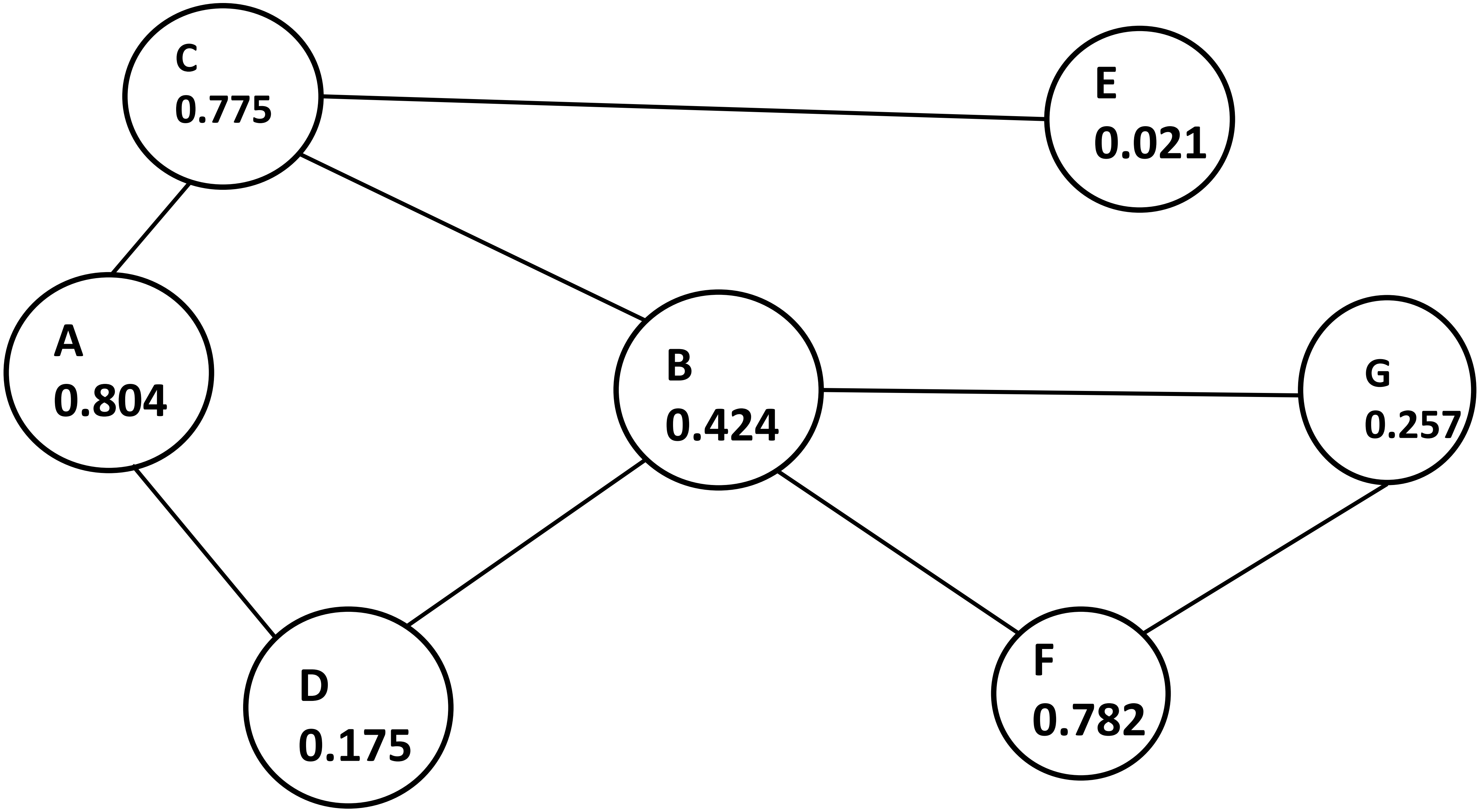
LOCAL SEARCH ALGORITHM:

The local search algorithm initially requires a feasible solution which are the dominating sets S1, S2, ..., Sm. Now, a swap is attempted between the nodes vi and vj which are two nodes of two different dominating sets Sk and Sl respectively. If Sk and Sl remain dominating sets and the sum of the lifetimes of Sk and Sl increases, then the attempted swap is made permanent. Otherwise, the swap is not made. In this way, the algorithm attempts swaps between every node of every dominating set where the two nodes are from two different subsets. The algorithm continues as long as lifetime improvements can be made by making new swaps.

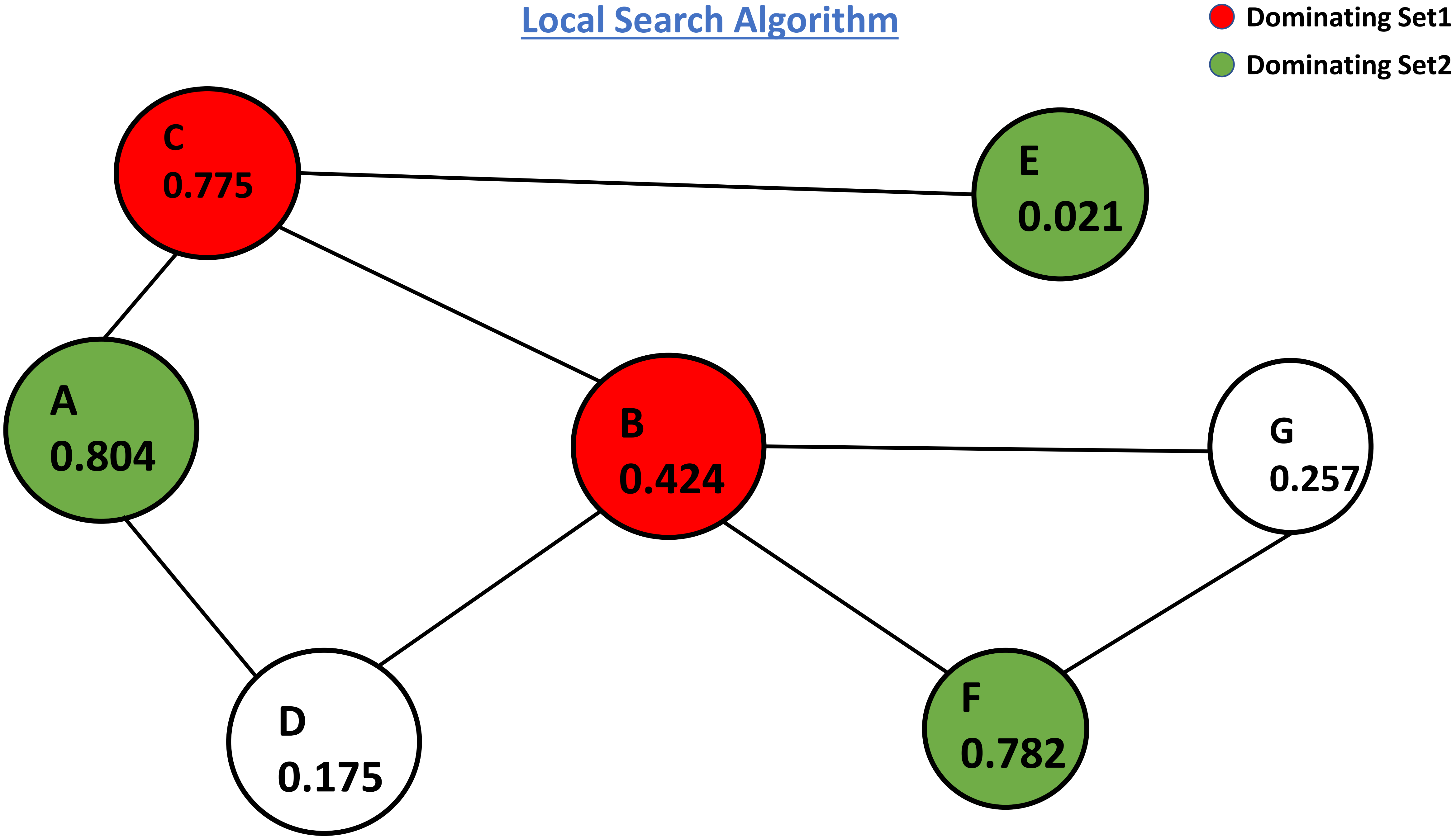
Implementation

A wireless sensor node is represented by a Python WSNnode class.
Some of the class fields are energy, the node id as well as a python list named neighbors.
It contains all the nodes within the communication range of the node.
If we apply the Domatic Partition algorithm we get 2 Dominating Sets. [C, B] and [A, E, F]

Local Search Algorithm

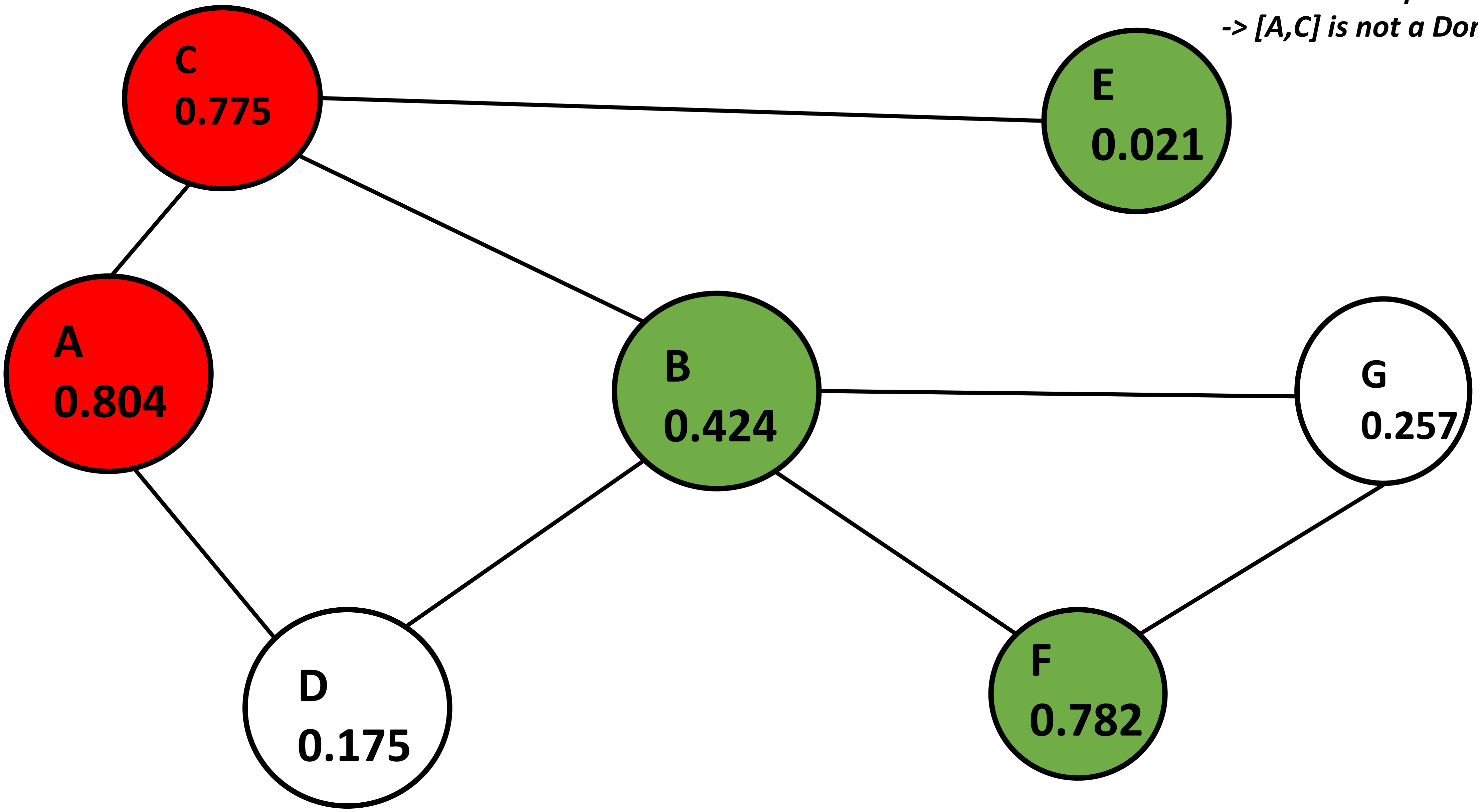


Local Search Algorithm



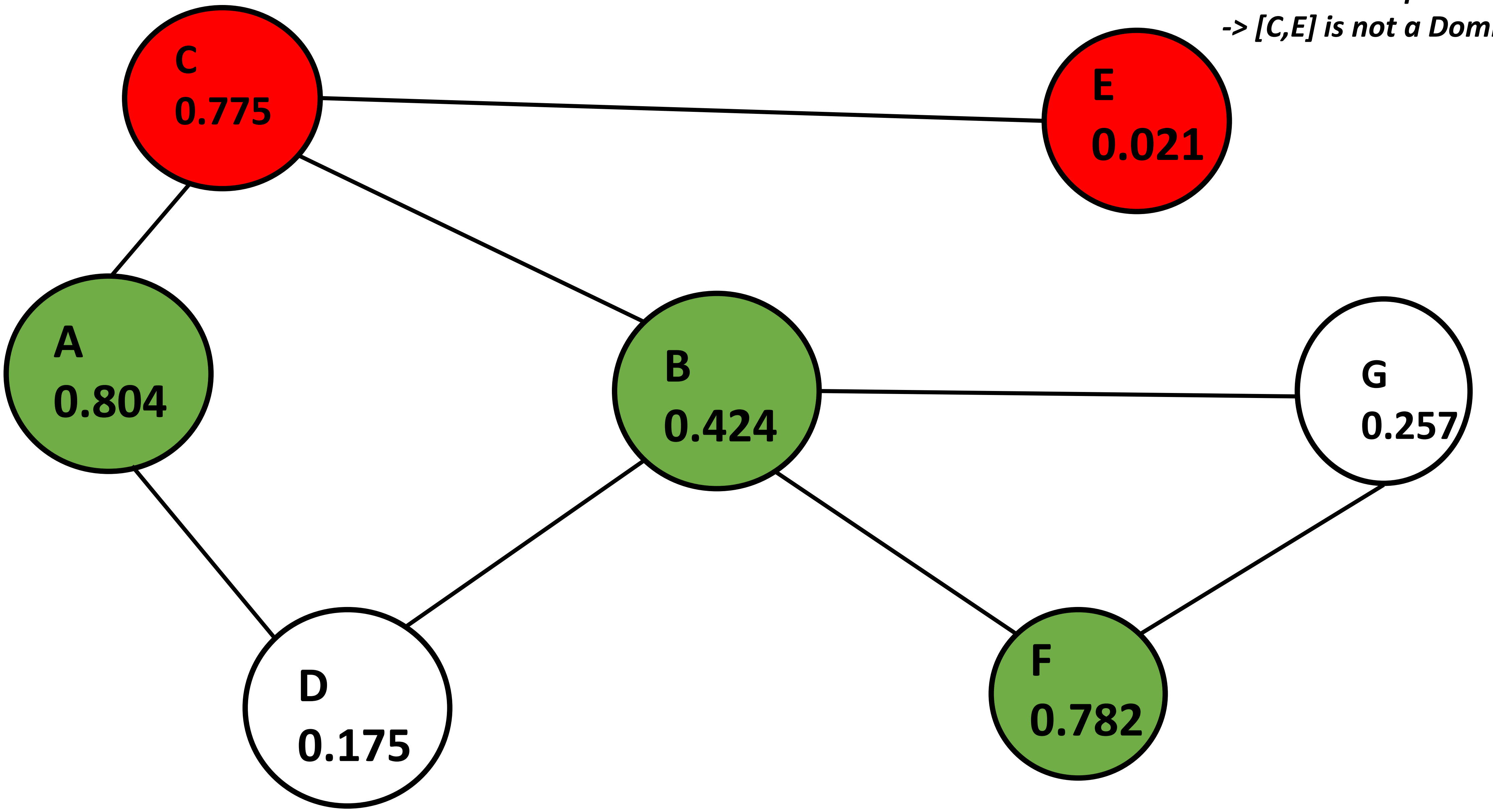
Local Search Algorithm

-> Swap A with B.
-> Cancel Swap.
-> [A,C] is not a Dominating Set.



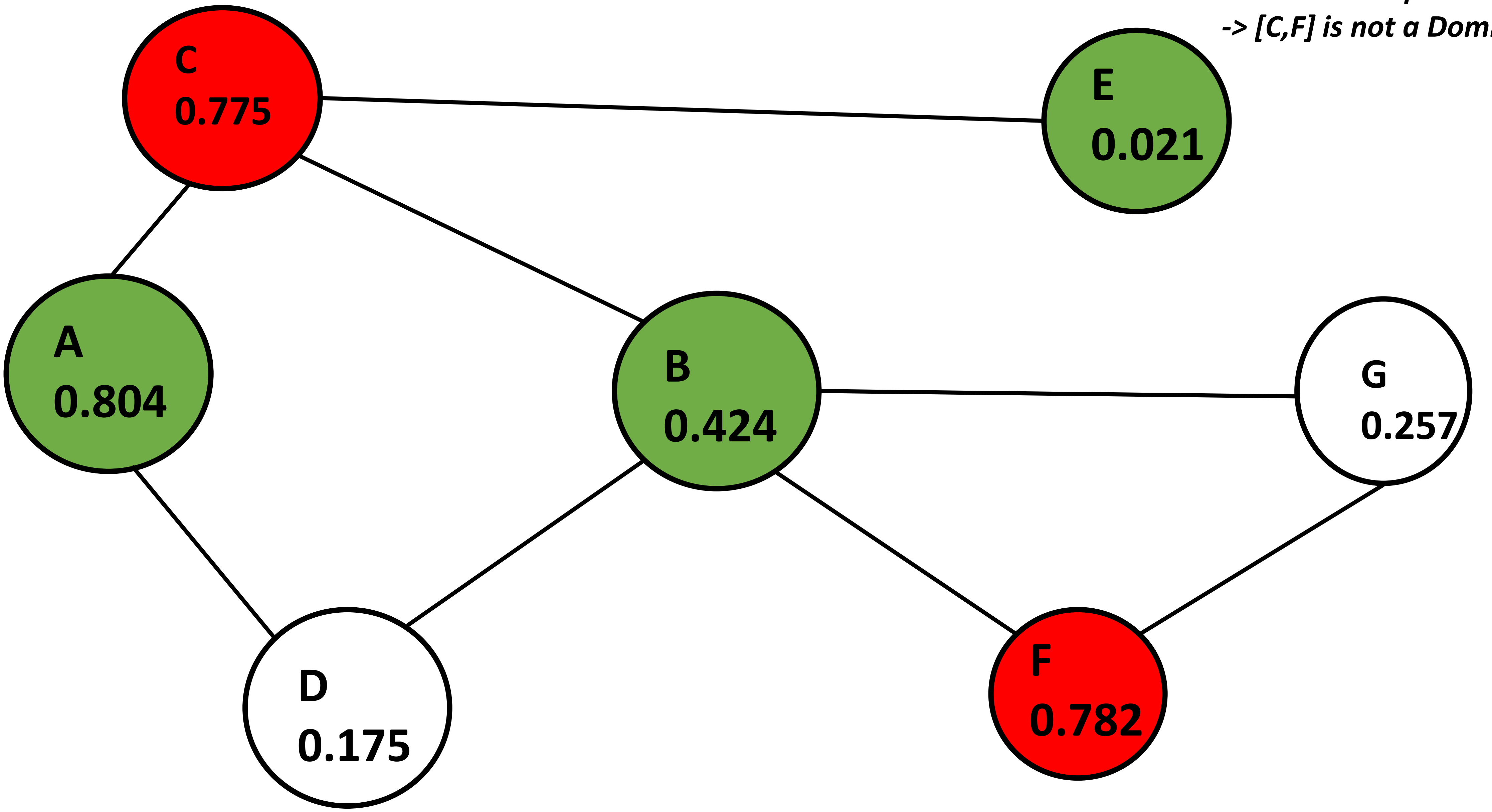
Local Search Algorithm

-> Swap E with B.
-> Cancel Swap.
-> [C,E] is not a Dominating Set.



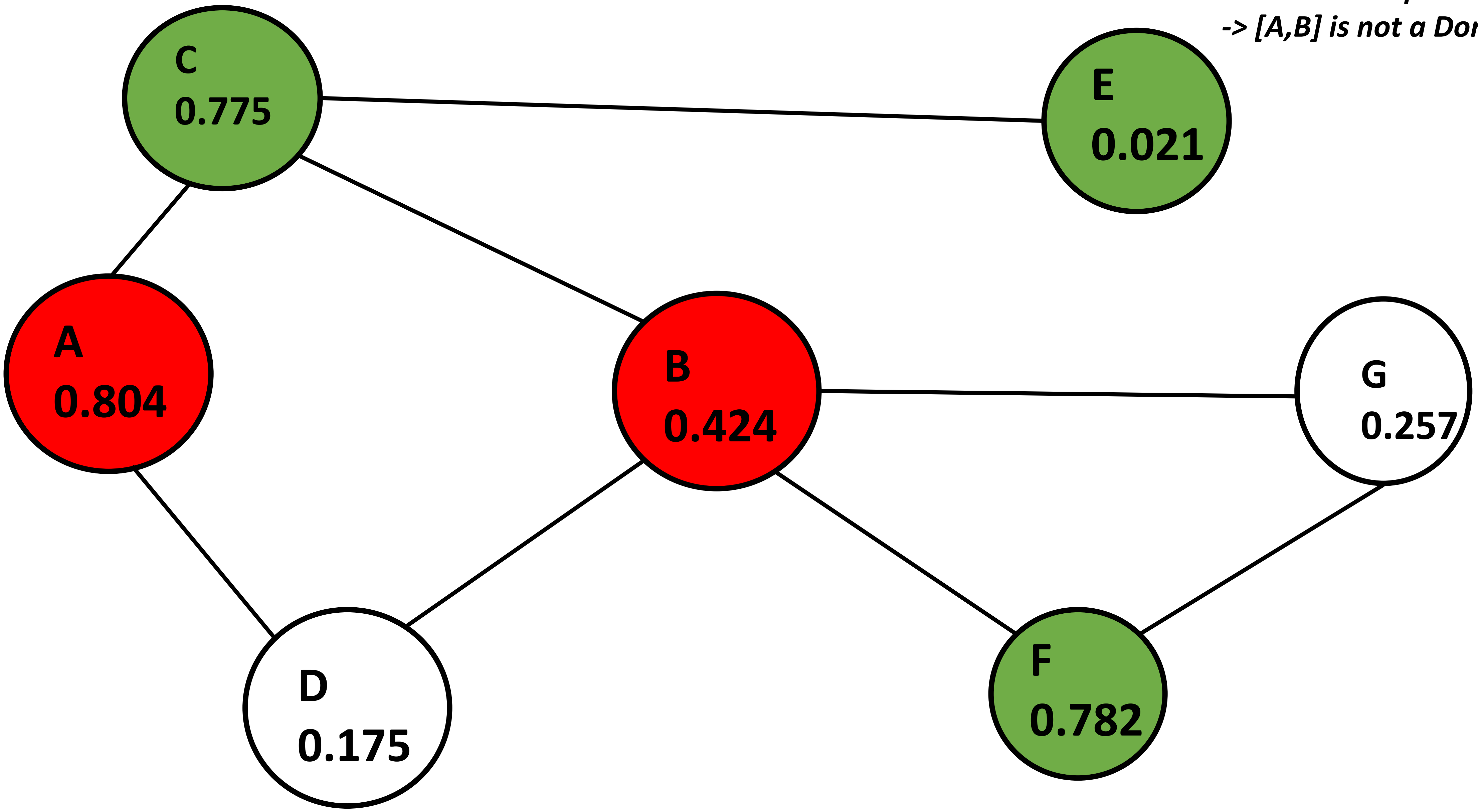
Local Search Algorithm

-> Swap F with B.
-> Cancel Swap.
-> [C,F] is not a Dominating Set.



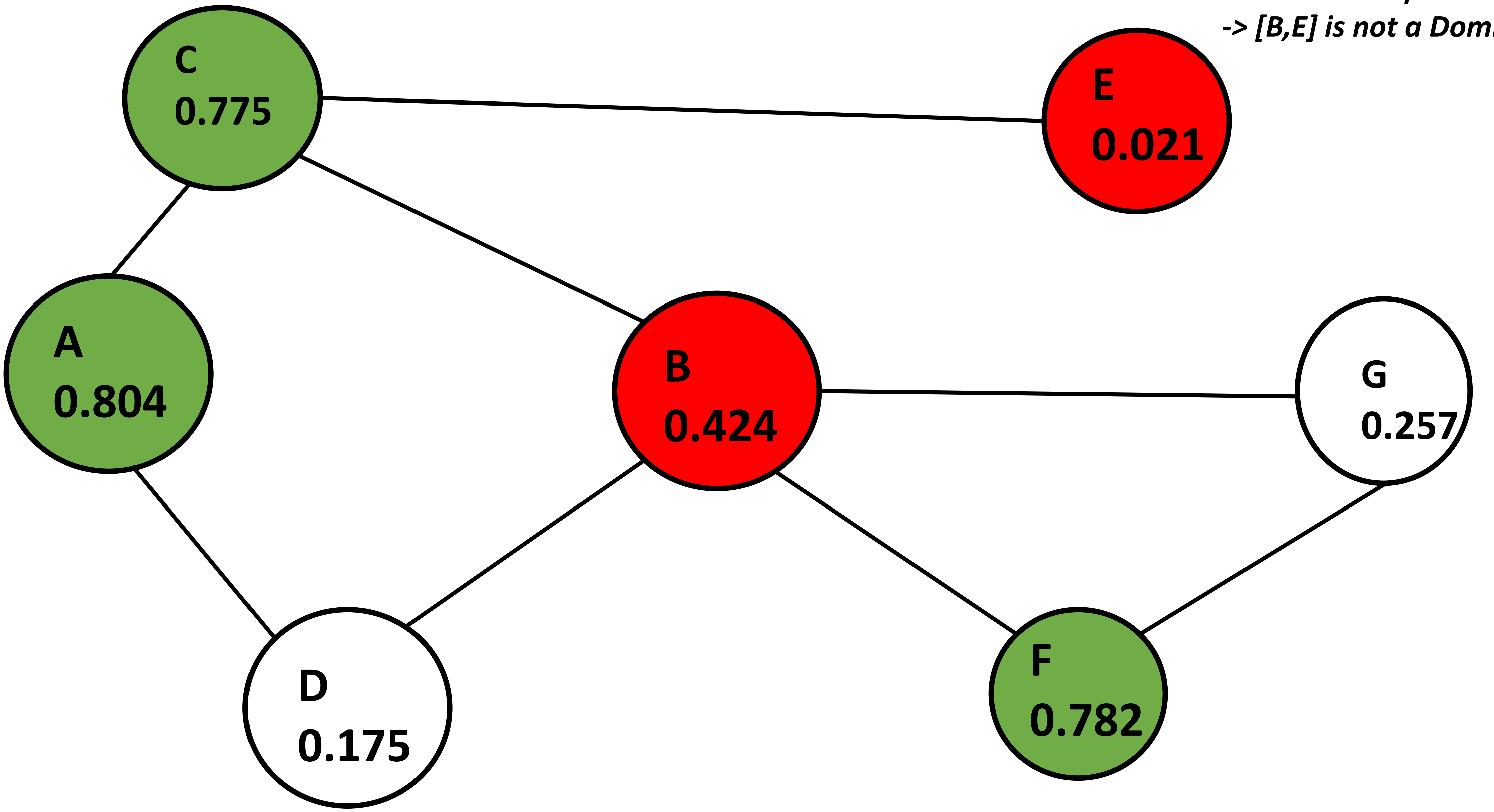
Local Search Algorithm

-> Swap A with C.
-> Cancel Swap.
-> [A,B] is not a Dominating Set.



Local Search Algorithm

-> Swap E with C.
-> Cancel Swap.
-> [B,E] is not a Dominating Set.

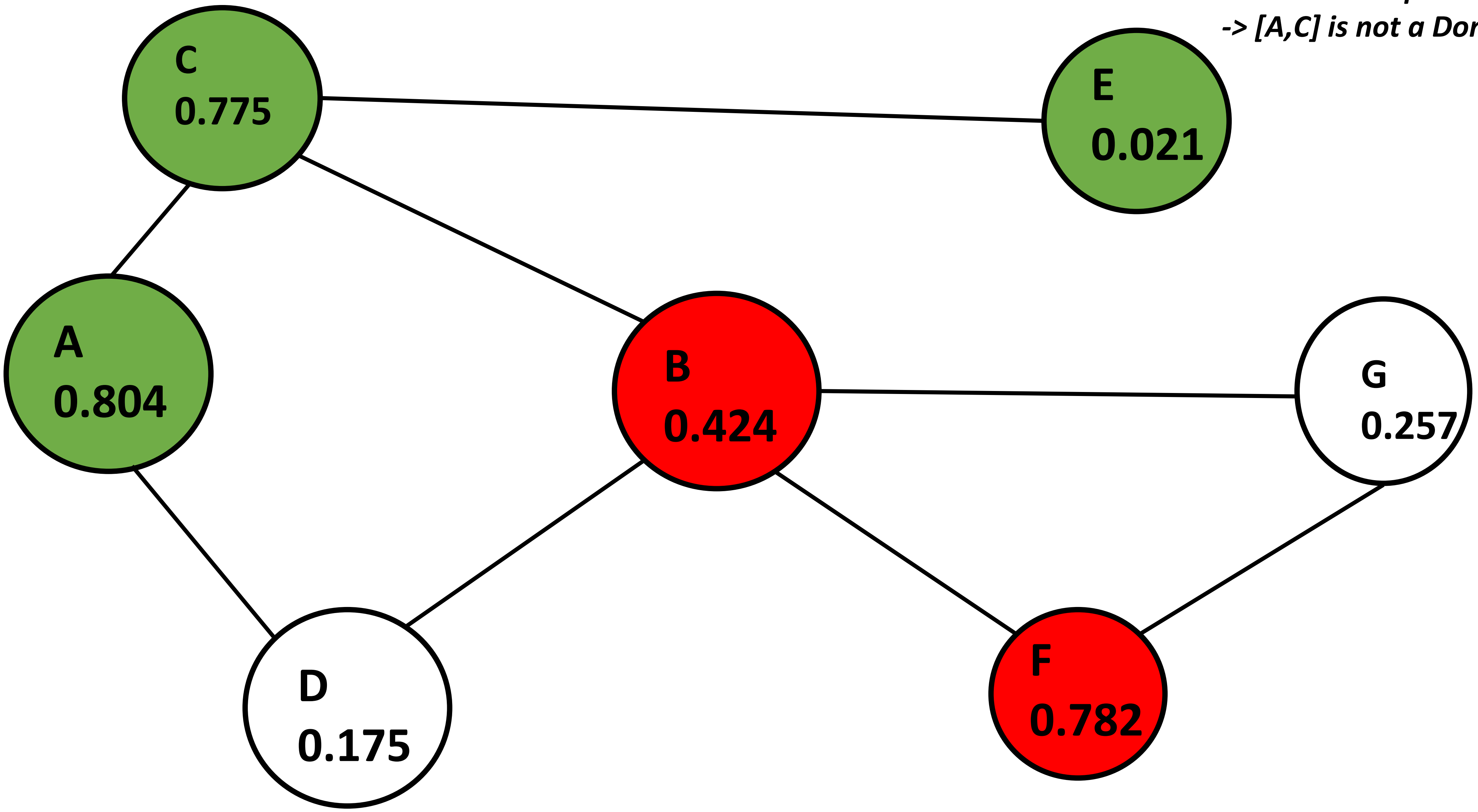


Local Search Algorithm

-> Swap A with B.

-> Cancel Swap.

-> [A,C] is not a Dominating Set.



tkasidakis@tkasidakis-HP-Pavilion-Gaming-Laptop-15-cx0xxx: ~/Desktop/mpc

File Edit View Search Terminal Help

```
[A]
[B]
Node [E] can communicate with:
[C]
Node [F] can communicate with:
[B]
[G]
Node [G] can communicate with:
[B]
[F]

 =====================
 ## Dominating Sets ##
 D(0):
 ['B', 'C']
 D(1):
 ['A', 'F', 'E']
 =====================
 [A] <-->[ENERGY]:0.8097275428200397
 [B] <-->[ENERGY]:0.014723645499226001
 [C] <-->[ENERGY]:0.5242780284593436
 [D] <-->[ENERGY]:0.10434675241461544
 [E] <-->[ENERGY]:0.26378033724957295
 [F] <-->[ENERGY]:0.623816311856386
 [G] <-->[ENERGY]:0.5380008284492789
0
=== New Temporary Dominating Sets ===
['A', 'C']
['B', 'F', 'E']
--> Swap B with A.
--> Cancel Swap
['A', 'C']
is not a Dominating Set


=== New Temporary Dominating Sets ===
['F', 'C']
['A', 'B', 'E']
--> Swap B with F.
```