The paper proposes a new scalable approximate algorithm (ACS) to solve the CSG problem which is proved to be NP-Complete. There are lots of different exact and approximate solution techniques in the literature for this problem. Most of them can solve the CSGP for less than 30 agents (i.e. they are not scalable to 100s of agents) and ACS is able to find a near optimal solution for hundreds of agents. Therefore, it is a novel technique in the related literature. ACS is designed heuristically to reduce the CS space which needs to be searched. Additionally, it encodes the nodes in integer partition (IP) graph such that each CS is represented as a list of integers in which the index and the element at that index represent the agent number and coalition it belongs to, respectively. Furthermore, ACS is approximate because it doesn’t consider some possible permutations (i.e. CSs) which is one of the methods to reduce the solution space.

Shortly, ACS works as following:

1. The integer partition (IP) graph represents the CSs based on the number of coalitions each node contains. First and last nodes in the graph are *Grand and Singleton Coalitions* which contain 1 and n coalitions, respectively*.*
2. An example node in IP graph is [1,3,6] which describes a CS and tells that the CS contains 3 coalitions which contains 1, 3 and 6 agents each. The ACS algorithm, encodes [1, 3, 6] as {0, 1, 2}. This is done for all nodes in the IP graph. For instance, [1, 4, 5] is encoded as {0, 1, 2}, [2, 8] as {0, 1}, [10] as {0} and etc.
3. The ACS computes the combinations of all encoded nodes. For instance, {0, 1, 2} has 3! combinations which are {{0, 1, 2}, {0, 2, 1}, {1, 0, 2}, {1, 2, 0}, {2, 0, 1}, {2, 1, 0}}.
4. Next step is to encode the combinations. For example, {0, 1, 2} => [0, 1, 1, 1, 2, 2, 2, 2] and let’s call this list as CS\_1. Each list represents a possible CS. The list contains 2 types of info which are 1) the agents and 2) the coalitions they belong to.