Finding Core Members of a Hedonic Game

Daniele M. Vernon-Bido
Old Dominion University, dvern001@odu.edu

FINDING CORE MEMBERS OF A HEDONIC GAME

by

Daniele M. Vernon-Bido
B.S. February 1987, Brooklyn College
M.S. May 1991, Boston University
M.S. December 2013, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Modeling and Simulation

OLD DOMINION UNIVERSITY
MAY 2022

ABSTRACT

FINDING CORE MEMBERS OF A HEDONIC GAME

Daniele M. Vernon-Bido
Old Dominion University, 2022
Dr. Andrew J. Collins

Agent-based modeling (ABM) is a frequently used paradigm for social simulation; however, there is little evidence of its use in strategic coalition formations. There are few models that explore coalition formation and even fewer that validate their results against an expected outcome. Cooperative game theory is often used to study strategic coalition formation but solving games involving a significant number of agents is computationally intractable. However, there is a natural linkage between ABM and the study of strategic coalition formation. A foundational feature of ABM is the interaction of agents and their environment. Coalition formation is primarily the result of interactions between agents to form collective groups. The ABM paradigm provides a platform in which simple rules and interactions between agents can produce a macro level effect without large computational requirements.

This research proposes a hybrid model combining Agent-based modeling and cooperative game theory to find members of a cooperative game's solution. The algorithm will be applied to the core solution of hedonic games. The core solution is the most common solution set. Hedonic games are a subset of cooperative games whereby agents' utilities are defined solely by a preference relation over the coalitions of which they are members. The utility of an agent is non-transferrable; there can be no transfer, wholly or in part, of the utility of one agent to another. Determining the core of a hedonic game is NP-complete.

The heuristic algorithm utilizes the stochastic nature of ABM interactions to minimize computational complexity. The algorithm has seven coalition formation functions. Each function randomly selects agents to create new coalitions; if the new coalition improves the utility of the agents, it is incorporated into the coalition structure otherwise it is discarded. This approach reduces the computational requirements.

This work contributes to the modeling and simulation body of knowledge by providing researchers with a generalized ABM algorithm for forming strategic coalition structures. It provides an empirically validated model based on existing theory that utilizes sound mathematics to reduce the computational complexity and demonstrates the advantages of combining strategic, analytical models with Agent-based models for the study of coalition formation.

This dissertation is dedicated to my children, Bido and Sterling. As I tried to set an example for them, they became an inspiration to me. Together we can overcome anything and achieve everything.

ACKNOWLEDGEMENTS

This journey was definitely a marathon not a sprint. But it would never have happened with the strength and guidance I received. My thanks to my committee members – Dr. Andrew Collins, Dr. John Sokolowski, Dr. Hong Yang, and Dr. Christopher Lynch. Your clear direction, flexibility, and willingness to help me succeed in this endeavor was exceptional and I deeply appreciate your help and support.

First and foremost, I would like to thank my advisor Dr. Andrew Collins. Dr. Collins stuck by me and motivated me to keep going to many times to count. Being a PhD student required relearning so many things I thought I knew. He taught me to write when I was stuck in my head. He taught me to read hundreds of journals and not lose focus. He has been both friend and mentor. And he dragged me kicking and screaming to the finish line. I can honestly admit, I would not be here without him. Andy, thank you for being the special person that you are, for sharing this journey with me, and for personally making it your mission to ensure I graduated. You have my eternal thanks and friendship.

I also want to express my deep gratitude to Dr. John Sokolowski. Dr. Sokolowski provided me direction and inspiration when I was just starting out as a master’s student trying to decide if this was right for me. He helped me find a path, an internship, and an advisor. All of which set me on a course that changed everything for me. Dr. Sokolowski, I am truly thankful to have met you and wish you all the best in your retirement.

Thank you Dr. Christopher Lynch and Dr. Hamdi Kavak. You have been with me every step of this journey. Your friendship and assistance, reviews and dry-runs, and co-authorship during this entire process has been immeasurable. Dr. Jose Padilla’s reminders about the value of taking time to think about the subject and the process helped me through many struggles. And to the many others that have provided guidance and assistance along the way, thank you.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

1.0 INTRODUCTION

Coalition formation is most often reviewed through the lens of cooperative game theory. However, there is a natural linkage between coalition formation and Agent-based modeling. A coalition can be described as “community of concerned agents, who, on the grounds of negotiation protocols, make a decision to cooperate in order to complete a certain task or to reach a certain goal” [1]. Agent-based modeling (ABM) is a frequently used paradigm for social simulation [2-4]; however, there is little evidence of its use in coalition formations. There are few models that explore coalition formation and even fewer that validate their results against an expected outcome.

Cooperative game theory is often used to study strategic coalition formation, but solving games involving a significant number of agents is computationally intractable [5]. There are different types of solutions for cooperative games and different types of hedonic games. One of the most common solution sets is the core by Gillies [6] and one of the most versatile set of games are hedonic games – those that are defined by an agent’s preference for a coalition. However, finding the core of a hedonic game, for a game of a significant size, is NP-Complete [7].

The ABM paradigm provides a platform in which simple rules and interactions between agents can produce a macro level effect without the large computational requirements. I believe it can be an effective means for approximating cooperative game solutions for large numbers of agents providing the theory can be adequately represented by the simulation code.

1.1 RESEARCH QUESTION

How can an ABM algorithm be designed such that its outcome is a member of the core solution set of a hedonic game greater than 90% of the time and the computation of the outcome completes in polynomial time?

1.2 PROBLEM STATEMENT

Coalition formation problems are prevalent in many disciplines. However, examining all possible coalition structures is computationally intensive. Coalition structures are the partitioned set of agents such that each agent in the set belongs to exactly one coalition and there are no empty coalitions [5]. The number of possible coalitions is given by the Bell numbers [8], the count of possible set partitions in combinatorial math. Table 1 provides a sample of the exponential growth of coalition structures to be explored as the number of agents increases.

Table 1: *Exponential growth of possible coalition structures based on the number of agents.*

# of Agents	**# of Possible Coalition Structures**
3	5
5	52
10	115,975
15	1,382,958,545
20	51,724,158,235,372

Cooperative game theory is an economics-based system derived from the work of Von Neumann and Morgenstern [9] and first described by Aumann and Drèze [10] that relies on rational decision makers forming coalitions based on utility it provides. Cooperative games are those in which the agents of the game "can make binding agreements about the distribution of payoffs or the choice of strategies…" [11]. Cooperative games have many possible structures; agents can decide how to divide their utilities in a coalition (transferrable utility), or they cannot

not allow utility transfers (non-transferrable utility). One specific type of cooperative game, hedonic games, is a non-transferrable utility game in which the agents' utilities are based on their preference for that coalition. Chalkiadakis, Elkind, and Wooldridge [5] note that any cooperative game can be described as a hedonic game.

There are different solution sets associated with cooperative games but the most commonly used one is the core solution set defined by Gillies [6]. Core members of a hedonic game represent a stable coalition structure. There is value in being able to find a stable coalition structure in the context of coalition formation. However, finding the core solution for all but the smallest number of agents is computationally taxing. The computation complexity of the coalition structure generation and comparisons using a naïve or "brute force" algorithm has a worst case magnitude of $O(n^n)$ [12]. In fact, determining the core of a hedonic game is NP-Complete [7] as is determining if a coalition structure is a core member [13].

NP-Complete problems, by definition, are not solvable except in trivial cases in polynomial time. However, Agent-based modeling and simulation (ABMS) is a technique that might be used to aid in finding a stable coalition structure. There are examples of models that have combined ABMS and Cooperative Game Theory but have not demonstrated the accuracy or sufficiency of their models with respect to finding a core stable coalition structure. Shehory and Kraus [14] combined cooperative game theory in hybrid models that determine coalitions for task allocation in multi-agent systems. Collins and Frydenlund [15] model refugee group formation and movement using an Agent-based model and the underlying principles of cooperative game theory.

Bonnevay, Kabachi, and Lamure [16] provide some insight into agent coalition rationality, but they neither consistently find a stable coalition structure nor provide a measurement for the frequency in which a stable structure is achieved. Collins and Frydenlund [15] use an adaptation of the concept of the core by having coalitions form based on the agents' utility of group size and speed; agents at each time step test a random coalition to determine if the new coalition improves the value for all member in the new coalition. Like Bonnevay et al., there is no measurement of how successful their algorithm is at reaching a core coalition structure. Janovsky and DeLoach [17] use a multi-agent simulation that allows agents to enter and leave coalitions based on the value of the coalition. While they compare their results to the current state of the art C-Link [18] system, their application does not provide a true representation of the core solution of self-interested agents.

Each of these models demonstrate the benefit of combining cooperative game theory with ABMS. However, none of the models provide any empirical validation against the defined cooperative game theory solution. My objective is to provide an Agent-based model that is capable of finding a core member greater than 90% of the time and validating that model through empirical statistical means. The 90% objective is a means by which we can determine the level of uncertainty that we are willing to accept. 90% is chosen as a compromise between the traditional 95% that is often used in engineering applications and the traditional rate used with Cronbach's [19] alpha of 80%. Cronbach's alpha is a measure of internal consistency. It should be noted that the results provided for the experiments will be given as a frequency measure that will allow researchers to accept or reject the level of error based on their requirements.

1.3 MOTIVATION

Although determining core stable coalitions, and cooperative games in general, are normally considered economic theory problems, coalition formation problems appear in many disciplines. Coalitions are temporary alliances of groups or individuals that combine resources and/or power for a particular aim [20]. The study of coalition formation and stable coalition structures is used in many disciplines. Political scientists such as Laver and Shepsle [21] and Bäck [22] examine the formation of governments and party coalitions. Studies of voting blocs [23, 24] strive to understand the possible combinations that will occur to secure ample voting or veto power to win an election or pass a resolution. In ecology, Silk et al. [25], Thierry [26], and Janson [27] study coalition formation among primates. Multi-agent simulations use coalition formation theories to allocate effectively tasks among different agents [14, 28-30]. Military coalition partners strengthen their ability to attack or defend by virtue of their combined resources and efforts [31]. Businesses form partnerships, communities form action groups, and consumers band together with the intention of achieving a goal [32].

Coalitions are a pervasive part of our social existence. The number of possible coalitions makes it is difficult to know which coalitions are likely to form, which coalitions will be profitable to their members, and which coalitions will be stable. The implications of coalition formation are critical to decision makers. The choice of coalitions can aid or hinder task completion. Poor choices can lead to loss of market share for businesses [33], insufficient military power [34], poor distribution of public goods and services, or political instability. The implications of coalition formation studies are broad and cross many disciplines. The technique in studying coalition formation, however, has most often been cooperative game theory.

Although cooperative game theory is the predominate technique for studying coalition formation, coalition formation has characteristics that make it a candidate for study using Agent-based modeling and simulation (ABMS). Agent-based modeling is a popular analytic method for the social sciences [35]. The Agent-based modeling paradigm is flexible with its inherent ability to demonstrate complex interactions through simple rules. It “concerns itself with modeling agent interactions… (1) who is connected to who and (2) the mechanisms governing the nature of the interactions" [36]. Coalition formation can be described as the various connections of agents and their interactions while the simple rules can be the algorithmic comparison based on cooperative game theory.

A hybrid Agent-based/Cooperative Game Theory (CGT) model has advantages and disadvantages. This type of model operates in a designated computation time. This allows the researcher to examine significantly large sets of agents without the taxing computational overhead. The model can be designed to operate for a predetermined number of "simulation ticks" and the resulting solution will always be the best option tested to that point. That is, it will discard any solution that results in a worse condition than the current. Further, the simulation model will always generate a coalition structure. While the core solution set may be empty, the simulation model will provide a "close" coalition structure – that is, it will find a coalition structure that was at least as core stable as any other coalition structure it encountered. The disadvantage is that it is impossible to know whether the coalition structure produced is completely core stable. That is, it is impossible to be certain if the ABMS produced a coalition structure that is a member of the core solution. This level of uncertainty exists in many ABMS due to the inherently stochastic and flexible nature of the models. There are validation techniques that can be used to understand and, in some cases, mitigate this concern.

1.4 APPROACH

This research advances the work of Collins and Frydenlund [37]. They present a heuristic algorithm that provides a series of routines that coordinate the rearrangement of coalitions using cooperative game theory mathematics to assess which coalitions are accepted. Based on this work, I created a heuristic hybrid algorithm that will provide a means for simulating strategic coalition structures, provide a statistical comparison of the algorithm to the mathematical framework, and demonstrate improvements in the computational time required by the mathematical computations and the original algorithm upon which it is based. The algorithm will be tested using a set of cooperative games known as hedonic games. Hedonic games, first named by Drèze and Greenberg [38], are those in which agents have a preference for belonging to a specific coalition and that preference is represented by a utility function [38]. Preferences cannot be sub-divided among coalition members; therefore, hedonic games are a subset of non-transferrable utility games (NTU) as defined by Aumann and Peleg [39]. All NTU games can be defined as hedonic through their utility function. As such, hedonic games represent a wide array of possible cooperative games.

I attempt to find a stable coalition structure of the hedonic game as defined by the core solution set. It is neither simple nor trivial to determine a stable coalition structure of a hedonic game. Finding the core solution of a hedonic game is NP-complete [7] and determining core membership of a hedonic game is also NP-complete [13]. A core coalition structure is one in which no coalition members have an incentive to defect to another coalition. Mathematically, this is determined by comparing every coalition possible for each agent against the coalition there are in within each coalition structure.

There are advantages and disadvantages in utilizing an ABM/CGT hybrid model. Agent-based models provide a platform that easily handles many heterogeneous agents. This allows for defining the individual preferences as utility functions of the agents. The ability to apply simple rules to individual agents allows for an algorithm that is linear with respect to the agents rather than exponential. Additionally, the simulation can determine the best solution tested at the end of each successful run. A disadvantage of ABM/CGT hybrid is the inability to properly identify if a steady state is reached. The system runs for a predetermined number of time steps or until no changes have occurred within a given time. However, this does not guarantee that no other changes would not have occurred given additional time, or the solution reached is in the core. The core could, in fact, be empty or the system may have found a local maximum and ceased to change states.

With this work, I aim to create a hybrid ABM/CGT model that finds a coalition structure that is reasonably comparable to those in a solution set. The model will incorporate the strategic nature of cooperative game theory into ABM and add to the modeling and simulation body of knowledge by creating a framework for merging ABM and cooperative game theory, providing a reusable model structure that applies to multiple types of games, and creating a validation structure that demonstrates the significance of the order of routine execution in strategic group formation.

1.5 CONTRIBUTION

This research is intended to advance simulation work in the field of coalition formation. It contributes to this field in the following manner:

- Provides a hybrid model that directly applies to cooperative game theory.
    - The heuristic algorithm created for the ABCG model is empirically validated to ensure consistency with the underlying theory.
- Provides a structure that efficiently stores and retrieves coalition values for comparisons.
    - Coalition values are calculated once and stored in a $2^N xN$ matrix.
    - Agent membership in a coalition is binary: 0 for not a member and 1 for membership. A binary string is created from members and converted to a

    decimal value. The decimal value is then used as the index for the matrix. This produces an efficient storage and retrieval mechanism for a large matrix.

- Provides a method for using Agent-based modeling in the study of strategic coalition formation.
    - The ABCG model is reusable for different cooperative game solutions. The model stores the coalition values in a matrix that is independent of how they are calculated, and the comparisons can be easily modified to solve for different solution concepts. For example, instead of individual comparisons of the core, the coalition values could be summed for a social welfare result with no changes to the interaction methods or the coalition value storage or retrieval.

1.6 ORGANIZATION

The remainder of this work is organized into four chapters. The next chapter provides a literature review of cooperative game theory solutions, hedonic games, and the association of ABM with cooperative games. This is followed by a chapter that describes the research method I used to explore and validate my solution for finding a core member of a cooperative game. The section begins with a description of the technique used to generate and solve the hedonic games. This is followed by a description of my Agent-based Cooperative Game (ABCG) model. Details are provided on the experimentation method followed by a discussion on validation techniques that will be used in this research. The next chapter provides a review of the experimentation results and analysis. It reviews the results of the experiments and the statistical significance of the efforts. The final chapter presents my conclusions and future work.

2.0 LITERATURE REVIEW

In this research, I examine the use of a heuristic algorithm to find a coalition structure that is a member of a core hedonic game. This chapter provides the reader with information in the literature that is pertinent to this concept. My heuristic algorithm is an Agent-based model (ABM). The first section provides general information about Agent-based models, their use and limitations in social science. The next section provides information on cooperative game theory (CGT), cooperative games, and cooperative games solutions. This is followed by a discussion on the current ways ABM and CGT have been used in concert. I review benefits and gaps of the current methods in the literature and how my research partially addresses that gap.

2.1 AGENT-BASED MODELING

Gilbert [35] defines Agent-based modeling as "a computational method that enables a researcher to create, analyze, and experiment with models composed of agents that interact within an environment." Macal and North [40] describe ABMS as an "approach to modeling systems comprised of autonomous interacting agents." The benefits of ABM include its ability to represent natural systems with non-linear behavior, its ability to capture emergent or unexpected phenomena, and its flexibility [41]. The focus of Agent-based modeling is on individual data [42] and individual agents rather than the collective during the modeling process. This allows researchers to use a "bottom-up" method of study. That is, rather than a central control or an aggregated structure, each agent produces their unique results based on their interactions with other agents and/or their environment.

Early research on Agent-based systems and multi-agent systems (MAS) referred to the structure as a society of agents that "interact together to coordinate their behavior and often cooperate to achieve some collective goal" [43]. This is reflected in various models and studies that have been propagated. Epstein and Axtell [44], for example, use Agent-based models to show how self-organizing agents distribute wealth through interactions with the environment and with other agents. Schelling [45], even before Agent-based modeling was named, showed how simple rules and individual decisions affected group formations. He demonstrated how segregation occurred with mild biases based on the environment and agent interactions. This "bottom up" approach of individual interactions is a well-documented way for researchers to study group formations. However, strategic coalition formation is not often considered through the use of Agent-based modeling. It is usually studied using cooperative game theory [46].

2.2 COOPERATIVE GAMES

Cooperative game theory is an analytic framework for exploring coalition formation by rational, self-interested decision makers [47]. Cooperative games are those involving more than two agents that can make binding agreements, explicit or implied, with respect to strategies and distribution of payoffs [5, 11, 48]. The cooperative game G consists of a non-empty set of agents, $N = [1, 2, \ldots, n]$ and a characteristic function $v: 2^n \rightarrow \mathbb{R}$ which maps each coalition $C \subseteq N$ to a real number $v(C)$ also known as the value of the coalition [5]. The characteristic function defines the game in terms of the value a coalition can achieve regardless of the actions of agents not in the coalition [5, 46]. The characteristic function game is defined as $G = (N,v)$.

There are three broad categories of games: transferrable utility games, non-transferrable utility games, and hedonic games. Transferable utility (TU) games are those games that allow for agents to offer side payments or other divisions of the payoff to secure agent cooperation in the coalition [9]. Non-transferrable utility games (NTU) are those games without side payments [39]. Hedonic games, using the terminology coined by Drèze and Greenberg [38], are a subset of cooperative games with non-transferable utilities that represent a specific form of coalition formation games where the total payoff for each member in each coalition is predetermined and known to all agents [49, 50]. They are unique in that the agents' utility is dependent upon the members of their coalition [50, 51]. Gamson [20] expresses this in a more generalized manner expressing the coalition value in terms of the payoff and a non-utilitarian strategy preference. In other words, each agent has an ordered or rank preference for all coalitions to which they are a party.

Game theorists are interested in the rational outcomes or solutions of these games. The outcomes of a cooperative game are defined as the set of coalition structures and feasible payoff vectors associated with the game – *(CS, X).* Solving a game is dependent upon the valuation of the coalitions, the payoffs, and the desired result.

2.2.1 SOLVING COOPERATIVE GAMES

Sandholm, Larson, Anderson, Shehory and Tohme [30], Tohme and Sandholm [52] and Rahwan [53] divide the function of solving cooperative games into three main activities: (1) coalition structure generation; (2) determining and optimizing coalition values; and (3) dividing the payoff. The coalition structure generation is a known combinatorial problem. A coalition structure, also known as a partition, is a set of disjoint coalitions whose union is $N = [1, 2, \ldots, n]$ and the intersection of any of the subsets is null. The number of partitions that exists for a set of N agents is given by the Bell number (for details see [54]). Exhaustively searching the entire space has a complexity of $O(n^n)$. Researchers have demonstrated the ability to solve some problems with about 21 agents but they generally limit it to approximately 15 agents [30].

The second part of solving a cooperative game, determining and optimizing coalition values, presents a different set of challenges. The coalition value, designated $v(C)$, expresses the expected benefit or outcome that results from the coalition's formation. Coalition value determination is the process by which a relative numeric value is assigned to each possible coalition. For every set of n agents there are $2^n - 1$ possible coalitions and an infinite number of ways to assign values to them dependent upon the problem at hand. Once all coalition values are assigned, optimizing the coalition values is dependent on how optimization is defined. One form of optimization is social welfare. Social welfare is "the sum of all the values of all coalitions" [5]. Social welfare considers the overall valuation of the coalition structure without regard to the individual agent.

Three ways of solving cooperative games when social welfare is the objective are linear programming, dynamic programming and Agent-based models. However, each method is computationally extensive, often requiring restrictions or limitations to handle the computational load. The basic linear feasibility program is as follows[1]:

$$x_i \geq 0 \; for\ each\ i \in N \; where\ x_i\ is\ the\ imputation\ of\ agent\ i$$

$$\sum_{i \in N} x_i = v(N) \; where\ v(N)\ is\ the\ value\ of\ the\ grand\ coalition$$

$$\sum_{i \in C} x_i \geq v(C) \; where\ v(C)\ is\ the\ value\ of\ the\ coalition$$

The number of constraints associated with this linear program is $2^n + n + 1$ and it yields a solution that maximizes social welfare.

Dynamic programming is another technique used to efficiently examine the solution space of coalition structure generation. It can solve the coalition structure generation problem in polynomial time, *($O(3^n)$)* [55, 56] but the memory requirements are extensive [28, 56]. It does, however, guarantee finding a maximized value. Shehory and Kraus [14] reduce the complexity

[1] Linear feasibility programming assumes superadditivity of the game. That is, the value of two disjoint coalitions ($C_1 \cap C_2 = \emptyset$) joining is greater than or equal to the sum of the values of the two coalitions: $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$

by creating an algorithm that restricts the maximum coalition size allowed. Sandholm et al. [30] utilize a coalition structure graph similar to Figure 1 and bound the search by level. They prove that they can provide a tight bound, but not an optimal solution, by searching only the bottom two layers.

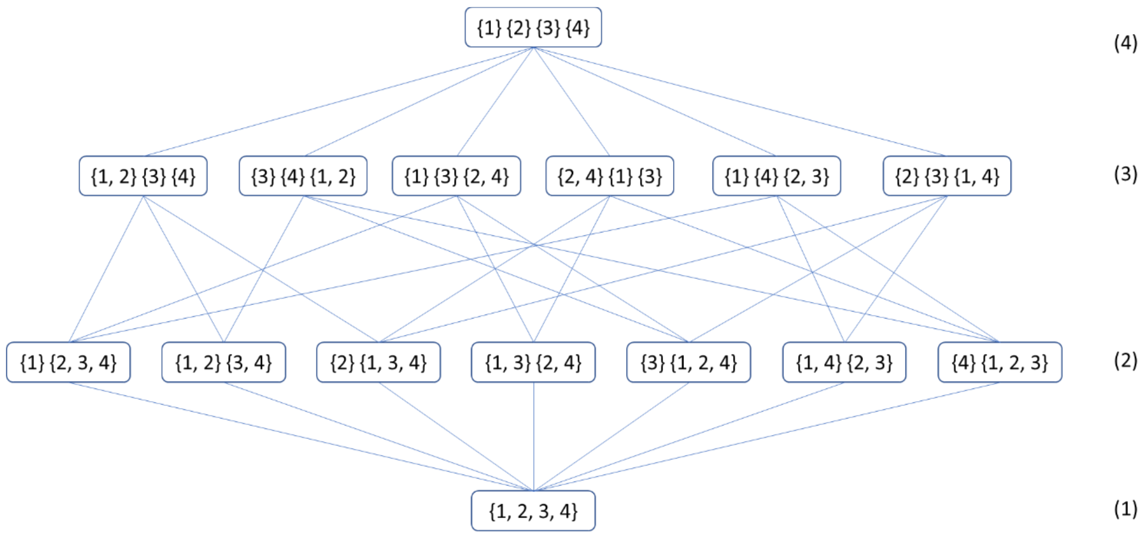


***Figure 1:** Coalition structure graph – each level indicates a coalition with one member added (if viewed from top to bottom) or removed (if viewed from bottom to top). Adapted from Sandholm et al. [30]*

Agent-based models are used to manage cooperative tasks rather than necessarily finding the optimum result. Shehory and Kraus [14], for example, show a multi-agent system problem in which agents must cooperate to move blocks of varying sizes. Completion of each task has an associated value; each agent in the coalition adds a cost. The value of the coalition is then assigned as the difference between the value of task completion and the cost of the coalition. Ramchun et al. [57] also incorporate size minimization into the coalition value determination. Zolezzi and Rudnick [58] use a cost basis for coalition valuation.

Agents form coalitions to achieve common goals. Sometimes the goals seek to maximize social welfare. Other times agents are looking to maximize personal gain. In these instances, an important component is the distribution of the payoff the agents receive from the coalition formation. Maximizing social welfare is a common solution goal, especially when the objective of the agents is altruistic. However, selfish or self-interested agents seek to maximize personal gain. This leads to the third activity in solving cooperative games, dividing the payoff. Distribution payoffs are often the basis of the solution concepts for cooperative games.

2.2.2 SOLUTION CONCEPTS

The characteristic function game defines the value of the coalition *v(C)* and implies the possible outcomes of the game *(CS, X)* but does not define how the payoff vector *X* will be distributed among the agents. Two of the most common ideas for evaluating the distribution of the payoff are fairness and stability [5]. These concepts reflect agents' desire to remain in a coalition based on their payoff.

Fairness is defined as the level to which an agent's payoff reflects its contribution to the coalition. The logic follows that an agent will participate in a coalition that provides a fair share of the payoff. There are numerous recommendations for the division of a payoff vector amongst its members. Shapley [59] provides a method of dividing a coalition's payoff based on the agents' marginal contribution. The Banzhaf index [60] likewise employs the concept of marginal contribution but calculates it over only the coalitions of the game [5]. Brown and Frendreis [61] show that distributions based on proportionality are common – bigger agents receive larger portions. Crott, Freiburg and Albers [62] examine an equal share structure. One of the general assumptions of these distributions is that the payoff can be subdivided in any manner deemed appropriate by the agents or the game.

Stability reflects the incentive that an agent has to remain in a given coalition [5]. Stable outcomes are those in which agents have no incentive to leave their current coalition to form a new one. Several measures of stability outcomes exist; two of the most common are the core [6] and the Nash equilibrium [63]. The equilibrium point of a cooperative game is the set of mixed strategies such that each agent maximizes his payoff if the strategies of all other agents are held constant. Nash notes that every finite game has an equilibrium point, but is not always solvable.

Examples of stability solutions are the nucleolus, the kernel, the bargaining set and the core. The nucleolus finds a coalition that minimizes dissatisfaction. That is, it selects the coalition based on "the difference between what the agents can get by themselves and what they actually get" [46]. The kernel is an equilibrium set in which no agent can reasonably expect a portion of another agent's payoff to form a new coalition [5]. The bargaining set examines objections and counter-objections to coalitions. That is, an agent *(i)* can object to the inclusion of another agent in the coalition *(j)* if all members of the coalition improve their reward without agent j. Agent (*j)* can counter if there exists a coalition with an agent (*j)* and without agent (*i)* such that the rewards are better for all members. If a counter-coalition does not exist, the original objection is said to be justified. The bargaining set is the set of all coalition structures in which no agents have a justified objection to anyone else in the coalition [46].

The core, originally defined by Gilles [6], is the oldest and the most widely used referenced solution set. The core of a game is based on dominance. Dominance occurs when, for a agent *i*, the payoff *i* receives in coalition C_a^i is strictly better than the payoff *i* receives in coalition C_b^i for all agents in the coalition[2] [46]. The core is the set of all coalition structures for which the payoffs of the coalitions cannot be dominated [46]. Chalkiadakis et al. [5] formally define the core as follows: "The core *C(G)* of a characteristic function game *G = (N,v)* is the set of all outcomes *(CS, x)* such that $x(C) \geq v(C)\ for\ every\ C\ \subseteq N$." Core sets are stable in that there is no incentive based on payoffs for any agent to form a new coalition. The core, however,

[2] C_x^i represents a coalition C_x that contains member *i.*

presents numerous challenges. While it is mathematically simple, finding the core is computationally complex. Additionally, the core is not guaranteed to exist; in some instances of games, the core is empty.

Fair and stable coalitions tend to endure long enough to accomplish the desired goals. While there are numerous ways to define fair and stable, for this research, I focus on the core.

2.2.3 DETERMINING THE CORE

Determining the core in these games is computational complex. The number of possible coalition structures for as few as 20 agents is over 51 trillion[3] [29]. The naïve approach to determining the core tests each coalition containing agent i against the portion of the coalition within coalition structure containing i, CS_i, to determine if there exist a coalition, C, in which every $i \in C$ has a strictly improved utility. Existence of such a coalition means that the coalition structure is dominated and therefore not a part of the core. This method is computationally intensive and cannot be used for large numbers of agents. For example, if there were 10 agents, there would be 115,975 coalition structures and 1,023 coalitions requiring approximately 593,212,125 comparisons[4] to determine the core. With 20 agents the number of comparisons is in excess of $542\ x\ 10^{18}$.

The utility may be a function of the payoff or may be the preference for membership in the coalition. Cooperative games in which membership preference is the agent's utility are called hedonic games.

[3] With 20 agents there are 51,724,158,235,372 possible coalition structure.

[4] Each agent appears in half of the coalitions: 10 x 512 = 5120 comparisons for each coalition structure (512 x 115,975 = 593,212,125.

2.2.4 CORE OF HEDONIC GAMES

Hedonic games, as mentioned previously, are those games in which the utility or coalition value is dependent only upon membership in the coalition. Determining the core is NP-complete [7]; determining core membership of a coalition structure in a hedonic game is co-NP complete [13]. The nature of hedonic games is such that any other game can be broken down and represented by a series of hedonic games [5].

The outcome of a hedonic game is a coalition structure; that is, each agent exists in exactly one coalition. A coalition may be a group of agents or a single agent. These games are identified by the preference relation that agents have for one coalition over another. The relationship is denoted as $a \succ b$ – the agent strictly prefers a to b – or $a \succsim b$ – the agent prefers a at least as much as b. The preference relationship on a set of outcomes Λ is defined with the following binary relation: $\succsim \subseteq \Lambda\ x\ \Lambda$. These properties can be defined as follows: (1) completeness, (2) reflexivity, (3) transitivity [5].

1. Completeness: for every pair in the set, the preference for one element is greater than or equal to the other.

$$\forall \{\lambda, \lambda'\} \subseteq \Lambda, \lambda \succsim \lambda' or\ \lambda' \succsim \lambda$$

2. Reflexivity: the preference for every element is greater than or equal to itself.

$$\forall \lambda \in \Lambda, \lambda \succsim \lambda$$

3. Transitivity: for every triplet of elements, if the preference for the first element is greater than or equal to the preference for the second element and the preference for the second element is greater than or equal to the third element, then the preference for the first element is greater than or equal to the third element.

$$\forall \{\lambda_1, \lambda_2, \lambda_3\} \subseteq \Lambda, if\ \lambda_1 \succsim \lambda_2\ and\ \lambda_2 \succsim \lambda_3, then\ \lambda_1 \succsim \lambda_3$$

Hedonic games are the complete, transitive, and reflexive preference relationship over coalitions contain *i* and are denoted as follows:

1. $G = (N, \succsim_1, \dots, \succsim_n)$
2. $N = \{1, \dots n\}\ is\ the\ set\ of\ all\ agents$
3. $\forall i \in N, \succsim_i \subseteq N_i x N_i$

The outcome of the game is *(CS, x)* where *CS* is the coalition structure and *x* is the preference utility vector.

Agents join coalitions to improve the value that can be achieved [11]. The agents in a hedonic game are assumed to be self-interested and individually rational; they will seek to change coalitions if their preference utility is improved. These games do not lend themselves to the social welfare maximization solutions; rather, core solutions are those with coalitions that cannot be dominated. Dominance occurs when the payoff of one coalition is better than the payoff of another coalition for every member of the coalition. Agents in a dominate coalition have no incentive to change to any other coalition. A coalition structure *(CS)* is blocked if there exist some coalition *(C)* that dominates the coalition structure for all agents in the coalition. That is, "$coalition\ C \subseteq N\ blocks\ CS\ if\ C \succ_i CS_i\ for\ all\ i \in C$" [5][5]. The set of *CS* that are not blocked in a hedonic game is the core.

The core of a hedonic game is individually rational and contractually individually stable. Individually rational implies that no agent will join a coalition that does not provide at least a payoff equal to what it receives on its own. Contractual individual stability exists when no agent can move to a coalition that improves the payoff for all agents in both the coalition from which the agent departed and the coalition to which the agent joined [5].

[5] CS_i refers to each coalition *C* in coalition structure *CS* that contains agent *i*

As mentioned, finding the core if it exists, is an NP-complete problem. Researchers have attempted to use various heuristics and multi-paradigm models to approximate a solution. In the next section, I review hybrid modeling and the combination of ABM and CGT.

2.3 HYBRID MODELING

“A hybrid is the result of merging two or more components of different categories to generate something new, that combines the characteristics of these components into something more useful” [64]. Hybrid modeling is the combined application of simulation with other disciplines such as engineering, applied computing, and operations rsearch [65]. In this section, I review hybrid models that combine ABM and CGT. Combining applications aids in increasing the benefits or overcoming weaknesses of both applications [66]. I explore how hybrid models of ABM and CGT have been used to aid coalition formation studies and evaluate any gaps in the literature.

2.3.1 ABM AND COOPERATIVE GAME THEORY HYBRID MODELS

Bonnevay et al. [16] attempt to advance the study by addressing the static nature of game theory. They combine game theory and ABM to study dynamic coalition formation. Their objective is to understand the rationality that leads to an equilibrium game. Assumptions made in their model provide the agent with attributes and logic used to determine when to make a proposition for a coalition and whether to accept a proposition. The model is useful to understand agent characteristics that result in coalition formation under certain conditions. However, it strays from the strictest ideals of game theory and is not as generalizable.

Szilagyi [67] create an Agent-based model to simulate the El Farol Bar Problem [68] as an iterated cooperative game. The Bar Problem consists of several individuals independently determining whether to attend a bar on a given night. The parameters for the decision include the bar's capacity and the individual's belief of how many others will be in attendance. The reward (or utility) is based on attending when the bar is not overcrowded and not attending when the bar is overcrowded. Szilagyi's model treats the coalition structure as an iterative two-person game with individuals selecting strategies based on biases and history; the agent is agent 1, and everyone else is agent 2. Each agent has a probability of cooperating (i.e., attending the bar) or defecting (remaining at home). Probabilities are updated based on rewards/penalties from prior action, neighbors' actions, and agent personality. The simulation is run until stability, or a definitive pattern is achieved. While the model provides insight into self-referential expectations and possibly simple financial market models, it does not address key cooperative game concepts of fairness or stability inherent in coalition formation. The coalition is not the focus of the model.

Collins [69] also uses an ABM to examine the Bar Problem. His emphasis, however, is more on the strategic group formation aspect with the model addressing agents forming and deciding to cooperate (attend) as a coalition. He incorporates routines of group mergers, group splits, and individual defection. This provides a framework for comparing the value of the collective decision versus the individual. The limit of this approach to strategic group formation is the lack of ability to determine whether the coalition structures formed aligned with any of the known solution groups of cooperative game theory (e.g., the core).

Collins and Frydenlund [15] use a hybrid Agent-based/cooperative game theory model to explore group formation during refugee migrations. Refugees form groups dynamically that provide safety and assistance to their efforts and leave groups that progress too slowly for their satisfaction. The model incorporates human movement models with strategic group formation. Their approach functionalizes determining core stability of a coalition structure focusing on the agent's decision to remain with a group or leave based on their utility as a member of the group. Their example translates utility into the strength of pooled resources.

Each of these combination models successfully demonstrate the use of hybrid models. However, they do not adequately address the argument that simulation is not backed by the underlying theory because they do not validate their models against theoretical results.

2.4 LITERATURE GAP

Cooperative game theory presents a reasonable structured manner to study coalition formation; however, its usage is limited because the computational requirement grows exponentially with the number of agents. Various attempts to minimize the computational complexity in algorithmic solutions are limited to solving cooperative games for maximizing social welfare. However, the do not address the most widely used solution set, the core. Finding the core of a hedonic game is NP-complete. Agent-based models have been used to attempt to approximate a cooperative game solution. They are designed for handling interactions of large numbers of agents and the resulting non-linear dynamics but have limited focus on the strategic component. While there are several coalition formation ABMs based on cooperative game theory, they are not validated against game theoretic results. The research gap in the literature demonstrates the need to develop a heuristic that produces results comparable to those found through analytic means that is less computationally intensive and is flexible enough to be useful in multiple applications.

This research intends to demonstrate the effectiveness of the existing algorithms in finding a coalition structure that is a member of the core and provide an alternative that is more effective and efficient. The benefit of this work is to provide researchers with an embeddable ABM algorithm to explore rational coalition structures of larger groups.

2.5 SUMMARY

In this chapter I provided a brief overview of ABM and its flexibility with respect to coalitions and collective interactions. I discussed the basics of cooperative games and cooperative game theory including the three primary types of games: TU games, NTU games, and hedonic games. I noted that a series of hedonic games can be used to represent any of the other types of games. I described the primary steps for solving cooperative games and reviewed some of the common solution concepts including the core and the computational challenges associated with solving cooperative games. After describing cooperative games and the solution challenges, I defined hybrid modeling and showed the use of Agent-based/cooperative game theory hybrid models. Finally, I discussed what I believe is currently a gap in the literature.

Cooperative game theory presents a reasonable structured manner to study coalition formation; however, its usage is limited because of the computational expansion that occurs as the number of agents increases. The three main functions of solving cooperative games: coalition structure generation, coalition value determination and optimization, and dividing the payoff; requires intensive computing resources. Researchers have found success in solving for larger numbers with less computational requirements when maximizing social welfare. However, there has been less success when selfish agents and alternate solution sets are considered.

The most common solution set is the core. The core is the set of coalition structures whose coalitions cannot be dominated by any other coalition. Finding the core of a hedonic game is NP-complete. Agent-based models have been used to attempt to approximate a cooperative game solution. They are designed for handling interactions of large numbers of agents and the resulting non-linear dynamics but have limited focus on comparisons to the underlying theory. While there are several coalition formation ABMs based on cooperative game theory, they are not validated against game theoretic results. The research gap in the literature demonstrates the need to develop a heuristic that produces results comparable to those found through analytic means that is less computationally intensive and can be validated through empirical comparisons to cooperative game theory.

The next chapter discusses the research method I use to create and validate the ABCG model which provides a direct translation of core of cooperative games.

3.0 RESEARCH METHOD

This section provides the rationale and outlines the steps of the quantitative research approach used in this dissertation. The objective of this section and more broadly the research is to evaluate clearly and concisely the effectiveness of the Agent-based Cooperative Game (ABCG) model in finding a core member for a cooperative game. I use a quantitative method to measure my results. “Quantitative research encompasses a range of methods concerned with the systematic investigation of social phenomena, using statistical or numerical data” [70]. In this case, the phenomenon is the ability to determine a member of a stable coalition formation as defined by Gillies as the core [6].

My research method consists of four major components. The first major component is generating the hedonic cooperative games. The hedonic cooperative game is defined by the following structure $G = (N, \succsim_1 . \dots \ \succsim_n)$, where $N \ = \ \{1, \dots \ , n\}$ is the set of agents in the game. The preference relationship “$\succsim_i$” denotes the preference that agent “i” has for a given coalition with respect to all other coalitions which contain that agent. The second component is solving the game for the core. I solve for the core by generating all possible coalition structures (*CS*) and eliminating any coalition structure (*CS*), that are blocked. “A coalition $C \ \subseteq$ $N \ blocks \ CS \ if \ C \succ_i \ CS_i$ for all $i \ \in C$ [5].” The third component is finding a core member of the hedonic cooperative games using the ABCG model. The final component is the validation of the model through statistical measures. Figure 2 graphically depicts the high-level steps of my research method.

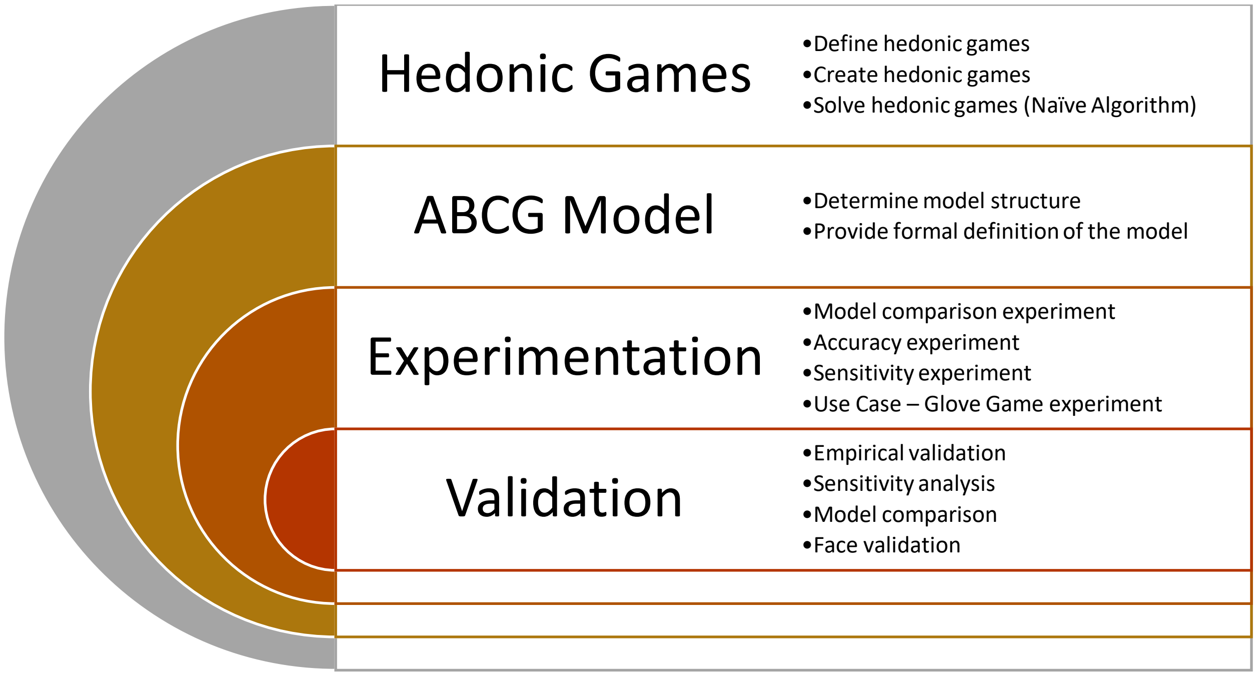


Figure 2: *Dissertation Research Method*

I then create a C++ program[6] to generate hedonic games and solve them for the core. Details of the program are described in the next section. The program is effective but not efficient in finding the core solution of a cooperative game. It employs a naïve or "brute-force" algorithm that checks every coalition structure against each coalition to determine if a coalition structure is blocked or a member of the core solution. Solving for the core of a hedonic game is NP-complete [7]. Computationally, it is infeasible to use this naïve algorithm for games consisting of large numbers of agents (approximately 17 [29]). The maximum number of agents I solve for in my program is 15 to allow for execution in a reasonable time frame. I estimate the limit in C++ for this program to be 27 agents since the maximum theoretical size of a 2-dimensional array in C++ is 4GB or 2^{32}. However, the practical execution with the memory drain will most likely be less.

[6] The first attempt at executing the naïve algorithm was done in Python 3.7. Python was unable to execute for greater than 12 agents.

Next, I create a model that is a hybrid of ABMS and cooperative game theory (CGT), my ABCG model. The model attempts to find a core member for each of the games created and previously solved in the C++ program. As previously mentioned, ABMS is a bottom-up type paradigm that examines individual interactions without centralized coordination. Agents within the coalitions, at different times, can change coalitions by leaving a coalition, joining with another agent or agents to form a new coalition, joining two coalitions together, splitting a coalition apart, or removing individual agents from a coalition. New coalitions formed by the interactions that prove to be advantageous for all members of the coalition, that is every member of the coalition improves their position or individual value in the new coalition, remain; otherwise, members return to the original coalition.

I perform various experiments through simulation once the model is completed. The initial conditions of the model are the array of values assigned to the coalitions determined when the games were created and the initial coalition structure. There are a series of four sets of experiments. The first set of experiments provides for the comparison of my ABCG model to the existing Collins-Frydenlund model. The objective is to determine if my model performs at least as well as the current state in the literature. The second set of experiments are designed to determine the accuracy with which the model can reach find a coalition structure that is a member of the core solution. The third set of experiments examines the effect of the initial coalition structure on finding a coalition structure that is a core member. The fourth allows me to demonstrate the effectiveness of the model in a known research use case – the glove game. Each set of experiments is designed to validate my ABCG model.

With the structure of these experiments, I have addressed for my model a common criticism of Agent-based models. Agent-based models have been criticized for lacking empirical validation [71]. I validate the ABCG model using several methods including empirical validation. With the first experiment, I validate both through model comparison and empirical testing. The original model and my ABCG model are tested against a known result to determine the frequency with which each model is able to reach the desired state. The second experiment expands the data used in the model and is again validated using a statistical comparison of the model result to a known result. The third experiment allows me to perform a sensitivity analysis to help me understand the brittleness of the model with respect to the initial coalition structure. The fourth, although empirically validated, is designed to show the model's usefulness in a common research problem. The problem is defined as the glove game. It is a simple market problem based on the assignment game of Shapley and Shubek [72]. These experiments provide for different subject matter expert (SME) face validations.

The remainder of the chapter is formatted into the four major sections. The next section will describe the creation of the hedonic games that are the basis for my experimentation. This will include the description of the naïve algorithm used to solve the hedonic games. That section is followed by a section describing my ABCG model. Section three contains the details of each of the experiments and the specifics of each of the validation technique. These major components were combined to provide a more comprehensive view of how the experiment and the validation technique are connected. Section four closes the chapter with a summary of the research method used in this dissertation.

3.1 CREATING HEDONIC GAMES

Drèze and Greenberg [8] coined the phrase "hedonic coalitions" to refer to the personal preference that people have for belonging to certain groups or coalitions. Hedonic games are cooperative games in which agents have preferences over the coalitions that they can join [3].

The value an agent has for a given coalition in a cooperative game represents that agent's preference for belonging to that coalition with respect to all other possible coalitions in the game. This represents the value designation step in the coalition formation problem. Hedonic games belong to the class of non-transferable utility games but are most interesting because any cooperative game can be represented by hedonic games [3]. The preference for a coalition can be random, relational, or based on the division of resources among coalition members. This ability to designate a preference based on almost any criteria provides us the ability to represent any cooperative game as a hedonic game.

For experiments two and three, I utilize strict hedonic games without the loss of generality. Strict hedonic games are noted as $G = (N, \succ_1, \dots, \succ_n)$, where $N = \{1, \dots, n]$ is the set of all agents. For each agent, $i \in (N, C_1, C_2)$, the notation $C_1 \succ_i C_2$ indicates that agent i prefers coalition C_1 over C_2. Agents in these games must prefer one coalition over another; indifference or ties between coalitions is not permitted. Agents appear in exactly half of the possible coalitions; the total number of coalitions is 2^N, where N is the total number of agents in the game. Agents' preferences are randomly ordered and assigned a value based on position in the ordering. I build an array of coalition values for each agent in every possible coalition. For example, in a three-agent game – agents a, b, and c – there are eight possible coalitions. Each agent appears in four unique coalitions. Table 2, shown below, is an example of one possible coalition value array.

Table 2: *Strict hedonic coalition value array*

Possible Coalitions	Agents' Preferences a	b	c
(a)	2		
(b)		4	
(c)			2
(a, b)	4	2	
(a, c)	1		4
(b, c)		1	3
(a, b, c)	3	3	1

I use a hashing algorithm to randomly assign coalition values. A random number between 1 and ½ the number of possible coalitions, $\left(\frac{2^N}{2}\right)$ or 2^{N-1} is generated. The algorithm checks to see if that number has already been assigned. If it has, the random number is incremented by 1. If the new number is within range (i.e., less than 2^{N-1}), the new value is checked; otherwise, the random number is set to 1. This continues until an available value is found. Once an unassigned number is found, the number is assigned to the coalition value array and is marked as assigned. This process continues until each position in the coalition array is filled. To determine coalitions to which the agent belongs, the algorithm takes advantage of a property of the binary version of the numbers. A “1” in the agent’s position of the binary number of the coalition indicates that the agent is a member of the coalition. It is simple and efficient; however, several other techniques, for example sorting algorithms or simple randomization of the numbers, could be employed with equal or greater efficiency.

The pseudocode for this algorithm is shown in Figure 3. The program is written in C++ using Visual Studio 2017. The random number generator used is Mersenne Twister [73], a common general purpose pseudo-random number generator that should not repeat during the execution time of our model.

```
GenerateCoalitionValueArray (N)
  for n = 1 to N
      set bin_matrix[n] = binary(n)
  end for
  for y = 1 to N
     set value_used_matrix[ ] = 0
     for x = 1 to $2^N$
        if (bin_matrix[x,y] = 1)
          hash_value = randombetween[1, $2^{N-1}$]
          while (value_used_matrix[hash_value] = 1)
               if (hash_value $\geq 2^{N-1}$)
                  hash_value = 1
               else
                  hash_value ++
          end while
          set coalition_value_array[x,y] = hashvalue
          set value_used_matrix[hash_value] = 1
          x++
     end for
     y++
 end for
```

Figure 3: *Hashing algorithm for assigning coalition values*

Once the coalition value array has been created, I attempt to solve for the core. I use a naïve or brute-force method for solving the hedonic game. This method entails checking every possible condition. The next section describes the algorithm and code used to find the core of the hedonic games created.

3.2 NAÏVE ALGORITHM

Gillies [6] defines the core as the set of coalition structures in which no agent has the incentive to deviate from one subset to another. The core, as defined by Chalkiadakis et al. [5], is the set of all outcomes (CS, x) such that $x(C) \geq v(C)$ for every $C \subseteq N$ where $x(C)$ represents the value of the current coalition for agent x and $v(C)$ represents the value for the agent in other possible coalitions. If there exists a coalition, $C \subseteq N$, for which the value an agent receives in that coalition is an improvement over the value it receives in its current structure, that current structure is deemed to be “blocked”. The core solution is the compilation of all coalition structures which are not blocked. Figure 4 shows the pseudocode for realizing these comparisons to determine blocked coalition structures.

```
CheckCoreCS(CS)
  # CS is the list of coalition structure
  # C is the list of coalitions
  # CS.value[ ] is the list of values for each agent in the partition
  # C.value[ ] is the list of values for each agent in the coalition

  set CS.blocked[ ] = 0
  for x = 1 to $2^N$
     set n[ ] = binary(x)
     for q = 0 to length(CS) – 1
          if CS.blocked[q] = 0
            set CS.core = TRUE
            for y = 0 to N – 1
              if n[y] = 1 then
             if C.value[y] <= CS.value[y]
                set CS.core = FALSE
                set y = N + 1
              end if
             y++
            end for
          end if
          if CS.core = TRUE
             CS.blocked[q] = 1
          q++
     end for
     x++
  end for
```

Figure 4: *Algorithm for determining blocked coalition structures*

The naïve or brute force algorithm approach [74] generates every partition and checks it against every coalition to determine if it is blocked. This method, while complete, is not elegant. The first step is to determine every possible coalition structure that exists for a given game. This task alone requires significant computations; the number of possible coalition structures for $n = 20$ agents exceeds 51 trillion [29]. This is followed by a comparison of every coalition to the coalition structure. The number of potential coalitions that can be formed are 2^n-1. Finally, a list of all partitions that have not been blocked is created [75]. This list of non-blocked partitions is the core. Maximizing social welfare, that is determining the greatest overall coalition value, has a complexity of $O(N^N)$ [29]; solving for the core increases this complexity to a level that makes the problem NP-complete. This limits the size of the cooperative game that I can reasonably solve. Rahwan et al. [29] solve for the social maximum and have managed to successfully compute games with 21 agents. Collins et al. [75] solve the core for a maximum of 13 agents. In line with the current literature, I solve the cooperative games for a maximum of 15 agents.

As mentioned, there are two main functions of the naïve algorithm employed: the creation of every coalition possible structure and the determination of which coalition structures are part of the core. The first function is a set partition problem. To solve this, I employ a method introduced by Djokić, Miyakawa, Sekiguchi, Semba and Stojmenović [76] known as setpart. The algorithm is shown in Figure 5 below. This method is functional but not necessarily the most elegant [75]. Expanding on the setpart algorithm would require further research into combinatorial set problems which is outside the scope of this work.

```
setpart(n);
  initialize
    c[n]
    b[n]
    r = 1
    j = 0
    n1 = n – 1
    c[1] = 1
    b[0] = 1
    continueLoop = true
```

```
while (continueLoop)
   while ( r < n1)
        r++
        c[r] = 1
        j++
        b[j] = r
   end while
   for j = 1 to n – j
        c[n] = j
         print out c[1] to c[n]
   end for
   r = b
   c[r]++
   if c[r] > n – j then j = j - 1
   if r = 1 then continueLoop = false
 end while
```

Figure 5: *Setpart algorithm from Djokić et al. [10]*

This algorithm generates a complete list of every possible coalition structure for a given number of agents. I iterate through this list as I compare the various coalition values to determine which coalition structures are blocked and which are not. This comparison is the second part of the program – the function to determine if a coalition structure is part of the core. The naïve algorithm systematically checks this condition for every possible coalition structure. Collins et al. [75] describe this function in the process diagram shown in Figure 6. The "Coalition Structures" box represents the input from the previous step. The coalition value array created in the game generation step contains the values for the "Coalition(S)" box and the values for the coalitions within the coalition structures.

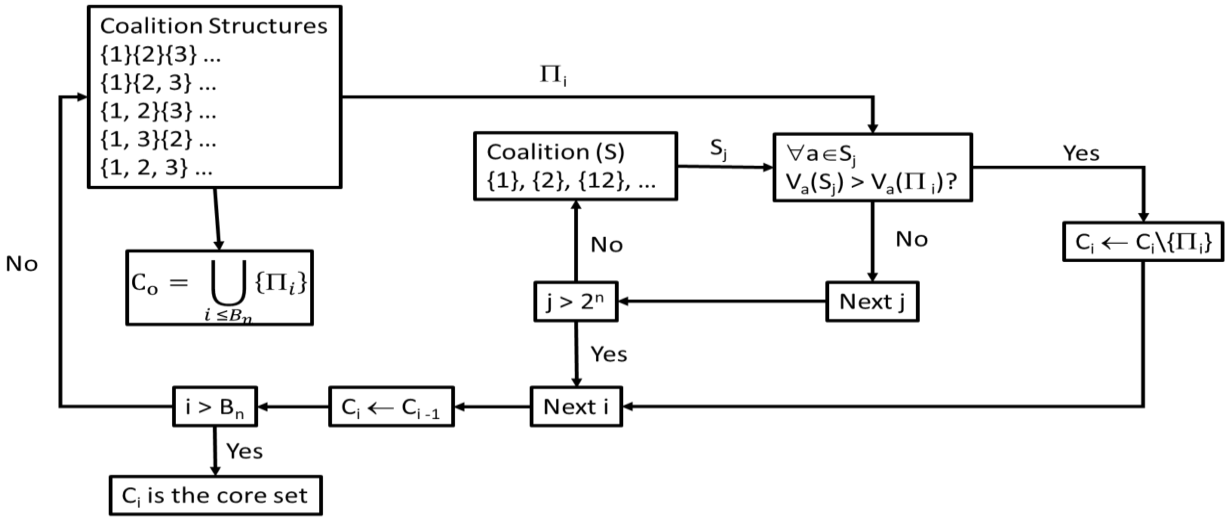


Figure 6: *Process diagram of naive algorithm for determining the core of a random hedonic game [9]*

The diagram shows the process of nested checks with the outer loop iterating over all possible coalition structures while the inner loop tests against all possible coalitions. Each pass determines whether a coalition structure is blocked. All blocked coalition structures are removed from the set; the coalition structures that remain at the end of all iterations is the core set. The algorithm is complete but probably not the most efficient.

Using the example shown previous of the coalition value array, Table 3 is an example of a 3-agent hedonic game with the core solution. I start with the coalition value array defined in the creation of the hedonic game. Each possible coalition structure is listed and the values for each agent in the coalition structure is detailed in the Agents' Values matrix. Agent "b" in this example will always prefer to be in a coalition by itself (coalition value = 4). Therefore, the only coalition structures that are not blocked due to "b" are [(a) (b) (c)] and [(a,c) (b)]. Agent "a" prefers to be in a single coalition (coalition value = 2) rather than a coalition with Agent "c" (coalition value = 1). Therefore, coalition structure [(a,c) (b)] is blocked. The only coalition structure not blocked is [(a) (b) (c)].

Table 3: *Example of 3-agent hedonic game*

Possible Coalitions	Agents' Preferences a	b	c
(a)	2		
(b)		4	
(c)			2
(a, b)	4	2	
(a, c)	1		4
(b, c)		1	3
(a, b, c)	3	3	1

Possible Coalition Structures	Agents' Values a	b	c
[(a) (b) (c)]	2	4	2
[(a,b) (c)]	4	2	2
[(a,c) (b)]	1	4	4
[(b,c) (a)]	2	1	3
[(a,b,c)]	3	3	1

Core Solution Set
[(a) (b) (c)]

My algorithm is a variation on this as it takes into account individual rationality coalition lists (IRCL) [7]. IRCL states that any coalition that provides an agent with less value than their singleton coalition, i.e., a coalition that contains only that agent can be ignored. I incorporate this into my algorithm using a pre-processing step that determines and marks coalitions that can be ignored during the iteration. While this increases the efficiency of the algorithm, there are still other techniques that could have potentially provided additional improvements. Further pre-processing beyond IRCL could be performed to eliminate the need to iterate other coalitions. Distributed processing rather than the current sequential looping could also have improved efficiency. However, I choose to simplify my efforts in creating the naïve algorithm. Additional pre-processing and distributed processing would increase the difficulty in assuring a verified solution. Further, the optimal efficiency of the naïve algorithm is outside the scope of this work.

My algorithm is implemented in Microsoft Visual Studio 2017 C++. The original work utilized Python 3.7 and Jupiter Notebook for solving the cooperative games with a naïve algorithm. Unfortunately, Python could not complete execution beyond 12 agents. Beyond 12 agents, the memory and space requirements for the comparison caused an error and an abnormal exit of the program. C++ allows for greater flexibility with memory and CPU usage. However, as shown later, the execution time required becomes extensive. The full code for the hedonic game creation and core solution is given in the Appendix.

3.3 ABCG MODEL

As seen in the literature review, there are several models that have combined ABMS and Cooperative Game Theory. However, very few provide a true representation of the core solution, and none have provided an empirical validation against the core. My objective is to create a hybrid heuristic model that provides a reasonable means for finding a core stable coalition structure. I determine that it is reasonable by empirically validating the solution against a known core solution.

I choose to start by using an existing algorithm. I feel that the model which most closely represents the cooperative game theory core solution is produced by Collins and Frydenlund [37]. The algorithm is clear and concise in its development of interaction routines. They describe 10 steps which are performed iteratively:

1. Select a random agent.
2. Randomly determine a new subgroup containing selected agent and determine the value of the subgroup as if it was independent of its group.
3. If the subgroup value is greater than the current group value, then subgroup detaches from the main group and move to step 9.
4. Determine the value of the group without the selected agent.
5. If group value is higher without the selected agent, kick selected agent out of the group and move to step 9.

6. Select another random local group (Moore neighborhood of a group agent). Determine if the selected agent's group benefits from merging with this random local group.
7. Determine if the random local group benefits from joining the selected agent's group.
8. If both benefit, group merge.
9. Randomly select a previously unselected agent and repeat from Step 2
10. If all agents have been selected, move to the next time step.

While the algorithm expresses dominance in terms of checking for the greater coalition value, it ignores individual rationality that is central to the core solution. I believe this creates a significant difference that would be demonstrated if the model were to be empirically validated against the core solution. I use this model as a starting point and improve upon it to more closely align my solution with core membership. I incorporate individual rationality into the model and ensure that I do not violate the properties of reflexivity and transitivity that are inherent in hedonic games. I believe these changes improve on the ability to find a core stable solution.

The ABCG heuristic model is designed to explore different coalition formations within a coalition structure. It is similar to the Collins and Frydenlund [37] in its efforts to find a core member through subset formation. Each new coalition formed is compared to the previous coalition to determine if all agents have a greater preference for the new coalition. My ABCG model incorporates the work of Collins-Frydenlund but improves on their model. Table 4 outlines the interaction differences. The ABCG model directly incorporates individual rationality and adds routines that create smaller coalitions to review the solution space more completely.

Table 4: *Comparison of model interaction methods between the CF model and the ABCG model*

MODEL INTERACTION COMPARISON	
Collins-Frydenlund (CF) model	ABCG model
Creating new coalitions	
Randomly select 2 coalitions to merge	Randomly pair agents to form coalition
Randomly select agents to form a new coalition	Randomly combine 3 or more agents to form coalition
	Randomly select 2 coalitions to merge
	Randomly select agent to joining different coalition
Dividing Coalitions	
Randomly split a coalition into two	Randomly split a coalition into two
Randomly remove an agent from a coalition	Randomly remove an agent from a coalition
Individual Rationality	
	Determine if agent is better off along

One of the main oversights of many of the Agent-based models that attempts to incorporate cooperative game theory is that selfish agents will not accept any coalition that does not provide them at least as great a utility as they have on their own. This is a primary consideration for determining the core of a hedonic game. To address this, I added a method that specifically checks for individual rationality. My algorithm also expands upon that approach by utilizing multiple methods of subgroup formation and verifying improvements at the individual agent level. Specifically, I utilize seven interaction routines to represent potential was coalitions can form. The interaction routines consist of uniting functions and separation functions. These provide the heuristic means for potentially forming new coalitions. Each agent’s coalition value is checked with every new coalition to ensure that every agent in the new coalition improves their position by adopting the new coalition.

The uniting functions – merge coalitions, pair coalitions, and trio coalitions – are techniques used to form new coalitions by bringing together two or more agents. The merge coalitions routine randomly selects two existing coalitions to create a single coalition. If every agent in the merged coalition improves their coalition value as a result of the merger, the new coalition is formed as part of the coalition structure. Pair coalitions randomly select two agents to form a new coalition. If both agents improve their coalition values by forming the new coalition, they depart from their existing coalitions and accept the new coalition as part of the coalition structure regardless of the impact on the remaining members of the old coalition. The trio coalition is like the pair coalition with three agents selected rather than two. The trio coalition is introduced to alleviate the local maximum potential. The algorithm used for my model belongs to a group known as "greedy algorithms." These algorithms are based on achieving locally optimal decisions with the hope of a global solution. The global solution is not actually represented or considered. Therefore, it is possible for a coalition to reach a value in which it is unwilling to reconsider given the rules for a change. That is, the game stalls at a local maximum. Early tests indicated that this occurred most often at the paired coalition level and that an additional rule could aid in preventing this condition.

There is also the potential for creating a larger coalition that is less than optimal. To minimize this problem, I have two the separation functions – exit coalition and split coalition. The exit coalition routine randomly selects an agent of the coalition to be removed from the coalition. If the removal of the agent improves the coalition values of all remaining agents, the agent is removed from the coalition and placed in a singleton coalition, that is, a coalition containing only the one agent. The split coalition function randomly divides the coalition into two coalitions. If in either coalition all agents have a higher coalition value than in the previous combined coalition, the coalitions are split into two.

The final two interaction routines focus on individual agent movement. The first, defect coalition routine, randomly selects an agent to leave their current coalition and join and new coalition. If the agent and all members of the new coalition improve their coalition value, the agent defects or leaves the existing coalition. The individual coalition ensures that all agents are individually rational. It checks each agent's current coalition value against their singleton coalition value. If the agent receives a higher coalition value being on its own, it leaves the current coalition. Table 5 shows a synopsis of each of the routines.

Table 5: *ABCG Routine Synopsis*

Routine Name	Routine Type	Description	Acceptance Criteria
Merge Coalitions	Uniting	Join two randomly selected separate coalitions	All agents improve
Pair Coalitions	Uniting	Join two randomly selected agents	Agents of the new paired coalition improve
Trio Coalitions	Uniting	Join three randomly selected agents	Agents of the new trio coalition improve
Exit Coalition	Separating	Remove a randomly selected agent from coalition	Agents remaining in the coalition improve
Split Coalition	Separating	Divide coalition into two	All agents of either new coalition improve
Defect Coalition	Both	Randomly select an agent to leave coalition to join another	All agents of the "defected to" coalition improve
Individual Coalition	Separating	Become singleton coalition to satisfy individual rationality	Individual agent improves

The selection of agents and coalitions is done using a Monte Carlo method. The Monte Carlo method, introduced by Metropolis and Ulam [77], is "a numerical method for solving mathematical problems by simulation of random variables [78]." In my hybrid model, agents are randomly selected to interact with other agents or coalitions. This stochastic aspect relies on a sampling of the potential solution space rather than the exhaustive comparisons demonstrated in the naïve algorithm. The advantage of this method is that it is significantly less computationally intensive, it always provides a solution set, and it is flexible in the number of iterations performed. The downside is that the solution is not guaranteed to be accurate – that is, the solution might not be a member of the core; this will always be the case if the core is empty – and the model produces only a single core member whereas the core set might contain several members.

The next major modification is the structural component. Solving hedonic games for the core is computation intensive even using a heuristic to approximate the value. My model gains efficiencies using two structural mechanisms. First, the coalition values are calculated prior to the first run and the model stores the set of coalition values in a matrix of size $2^N x\ N.$ This ensures that they only need to be calculated once. The coalition index number is derived from the agents in the coalition. An agent existing in a coalition is a binary function: 0 if they are not in the coalition; 1 if they are in the coalition. This binary string of agent membership in a coalition is converted to a decimal and is used as the index for the coalition value matrix. This use of binary properties allows the model to rapidly identify coalition values based on membership. Specifically, each coalition value is designated by the membership string and each agent is designated by position. This mathematical mapping reduces computational time required to compare the values of the new coalition to the existing coalition. It is set to derive or retrieve from a file the core values ensuring that the calculation of these values only occurs once. This part of the problem is reduced to $O(2^N)$ as opposed to completing the calculation for each comparison.

After each new coalition is formed, it is tested to determine if the new coalition improves the value for all members of the new coalition. The aim of this is to capture the functionality of the analytic algorithm that assesses if a coalition is blocked. If the new coalition that forms improves the value for all members of the coalition than the old coalition was blocked and is not a member of the core. Similar to Rahwan et al. [29], each iteration is guaranteed to be at least as stable as the previous since coalitions that do not improve the value for the agents are rejected. This also means that for every game, even if there is no core solution, a relatively stable solution will be presented by the algorithm. That is, while it might not be a core member, the coalition structure is at least as stable as all other coalition structures tested.

3.4 ODD PROTOCOL

The implementation of my model is done in NetLogo [79]. NetLogo provides a simplified coding language that incorporates the ability to randomize agent selections and schedule routines simply. Other ABM languages such as SWARM, RePast Symphony, or Mason could have served equally as well. One common challenge of modeling is a standard way to describe the model. One method that has grown in use is the Overview, Design Concepts, and Detail (ODD) Protocol [80]. The ODD Protocol provides a standardized method to describe Agent-based models [81]. It is generally used in social science. The next section provides a formal description of the ABCG model using the ODD Protocol.

3.4.1 PURPOSE AND PATTERNS

The purpose of the model is to generate coalition structures of different hedonic games, using a specially designed algorithm. The coalition structures are later analyzed by comparing them to core partitions of the game used. Core partitions are coalition structures where no subset of players has an incentive to form a new coalition. The patterns of coalition structures generated by the simulation are expected to converge to a core partition if one exists for a given game. The simulation is the model implemented over time.

3.4.2 ENTITIES, STATES, AND VARIABLES

The simulation model is a representation of a cooperative game theory games known as hedonic games. The hedonic game involves agents trying to be a part of their favored coalition. The focus of the game is which coalitions of agents form.

Environment: Abstract social environment where all agents are assumed to be able to communicate with each other with complete information.

State variables: All variables are associated with the players.

Table 6: *State variables of ABCG model*

Variable	Type, Range	Owner	Temporal
Coalition Membership	Integer, [0, # agents]	Agents	Dynamic
Coalition Value	Integer, [0, 2^{N-1})	Agents	Static

Coalition Membership: It gives the index number of the coalition that the agent is a member. If an agent is not a member of a coalition, it is assumed to be in a singleton coalition, and an index number is still assigned.

Coalition Value: This variable indicates the numbered preference an agent has for a coalition. Agents are members of ½ the total possible coalitions; the total number of possible coalitions is 2^N where N is the number of agents. The preference is strictly ordered with 1 being the least preferred coalition and 2^{N-1} being the most desired coalition. Note, this differs from several other definitions of coalition values in that it does not represent the composite of all members but rather the individual member values.

Scales: The temporal scales within the model are arbitrary. Each round represents an opportunity for several coalitions to be suggested to the agents and, if necessary, the updating of the coalition structure.

Process overview and scheduling: The scheduling of the game is such that there are several rounds (simulation ticks). Each round represents an opportunity for the agents to propose new coalitions, and, if acceptable to all potential members of that coalition, they form a new coalition. The agents' proposed coalitions are created by the algorithm.

The main loop of the simulation is as follows:

1. This is the algorithm subloop. The algorithm suggests seven types of coalition each turn. The coalition suggestions are discussed in the submodel section. At each step of this subloop, one of the randomly selected coalition types is suggested. First, all the computerized agents in the proposed coalitions are asked if they wish to join.
    - If any agent rejects the proposed coalition, then nothing changes, and the algorithm moves on to suggest the next type of coalition.
    - If all computerized agents agree to join the proposed coalition, then the proposed coalition forms and computerized agents update their membership. The algorithm moves on to suggest the next type of coalition

This subloop repeats until all seven suggested coalitions have been evaluated.

2. All agents' internal values are updated to reflect the new coalition situation if they have not already been done so. This includes all agents who have lost other agents because those other agents are moved to another coalition.
3. Loop to next round.

The critical point is that the algorithm will suggest several coalitions that are proposed to the agents. This algorithm is discussed in the sub-model section below. A flow diagram of the main loop is given below:

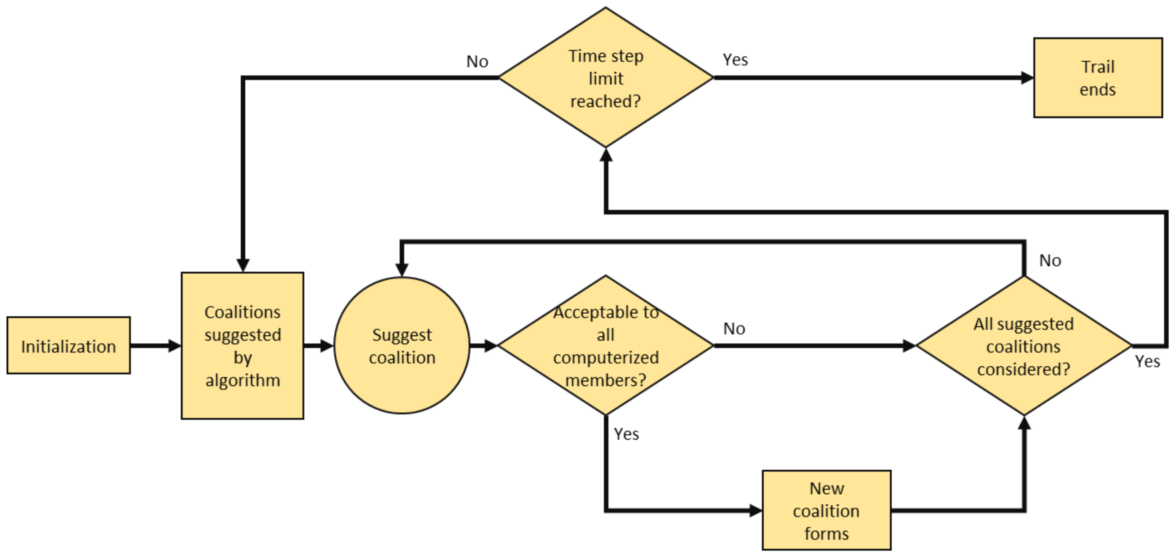

Time step limit reached?
No
Yes
Trail ends
No
Initialization
Coalitions suggested by algorithm
Suggest coalition
Acceptable to all computerized members?
No
All suggested coalitions considered?
Yes
Yes
New coalition forms

Figure 7: *Flow diagram of the main simulation process*

Design concepts

Basic principles

The underlying game of the model is a variation of the glove game, a classic game in cooperative game theory [82]. The computerized agents are assumed to be utility maximizers, which is consistent with game theory standards. Their utility is the sole driver for the computerized agent's decision-making, and complete information is assumed. The utility value in the experiments is equal to the coalition value.

Emergence

The main expected emergent behavior is that the final coalition structure, the collection of disjoint coalitions covering all players, is a core partition. A core partition is a covering set of disjoint coalitions where no subset of agents has an incentive to form a new coalition [50, 83]. It is related to the core concept, introduced by Gillies [6], but, strictly speaking, is not precisely the same. The core partition is an appropriate solution mechanism because it focuses on coalition membership instead of coalition values and imputations. A core partition is not necessarily unique for a given instance of a game nor is it guaranteed to exist. I only consider games with a non-empty core.

Adaptation

The ability of an agent to accept suggested new coalitions to join is the adaptive part of the model. The agents are only able to change to a coalition that is suggested to them by the algorithm. Further, a new coalition only forms if all potential members of that suggested coalition choose to join that coalition. This means that every agent in a proposed coalition has the veto power to stop the new coalition from forming. The agents will choose to join a new coalition if it increases their utility.

Note that agents might find themselves in a new coalition because other members of their coalition have decided to leave that agent's coalition. Agents cannot stop other members from leaving; they can only stop a new coalition forming that includes them. I do not assume a complete collapse of the remaining coalitions into singleton coalitions as others have [82].

Objectives

The objective of all the computerized agents is to join the coalition that maximizes their utility. The agent's utility is quantified as a reward. In the glove game, the reward is dependent upon the agent's preference. If agent 'a' prefers pairs of gloves, the utility for agent 'a' in coalition 'S' is:

$$R(a,S) = {min(\sum_{b \in S} L(b), \sum_{b \in S} R(b))}\Big/{|S|}$$

If the preference is for the number of gloves, the reward is:

$$R(a,S) = \sum_{b \in S} \frac{L(b) + R(b)}{2} \Big/ |S|$$

Learning

There is no learning incorporated into this model.

Prediction

There is no prediction incorporated in this model.

Sensing

There is no agent sensing incorporated in this model for the computerized agents.

Interaction

All agent interactions are mediated. That is, the agents do not directly interact with each other, but their actions do affect each other. These effects are due to the decision that they make with regard to coalition membership. If an agent leaves or joins a coalition, then the make-up of that coalition changes, which, in turn, affects the utility or coalition value of coalition members. The value of a coalition is determined by the members of the coalition and is different for each agent.

Stochasticity

The only stochastic element of the model is the random determination of which coalitions are suggested by the algorithm. The algorithm generates seven different suggested coalitions during this step of the model; each of the seven suggested coalitions is derived by a different coalition formation approach. An example of a coalition formation approach would be combining two randomly chosen coalitions to create a suggested coalition. The different coalition approaches are discussed in the submodel section below.

In all cases, the uniform distribution is used when selecting an agent or coalition. That is, all agents or coalitions will have the same probability of selection in a given coalition formation approach.

Collectives

The model has a focus on coalitions, which are a form of collective. The coalitions determine the utility that each agent would get, and, in turn, this utility drives the computerized agent decision to join any proposed coalition. The coalitions are explicitly represented in the model as a number; each agent has a coalition number assigned to it. Note that the set containing only one agent is still a coalition; it is known in cooperative game theory as the singleton coalition.

Observation

The final coalition structure is recorded after the game has completed 100,000 time-steps.

Initialization

All agents are assumed to start in a grand coalition, i.e., they start in a coalition of all agents. The number of agents and their coalition preference is determined by which game is being considered. Note that an agent’s coalition preferences do not change over the course of the game.

Input data

All variables are determined by the initial coalition preferences and the algorithm mechanics. This includes the random number generator that is used for the stochastic processes, which is determined internally by the simulation programming platform.

Submodels

There are three submodels used within the model: coalition selection, coalition evaluation, and coalition updating. These three submodels control the changes to the coalition structure, which is the main output of the model.

Coalition Selection

There are seven coalition suggestions (S) that are made at each round of the game. They are suggested in the order given below. A description in prose and mathematical notation is given for each. The seven suggestion types are:

Merge coalition

Two agents from different coalitions (U,V) in the current coalition structure (CS) are chosen randomly. The utilities of the merged coalitions are identified for each agent. If the utilities improve for all members of both coalitions, a new coalition is formed, which is the merged coalition. If the grand coalition (N) has formed, this suggestion is ignored.

$$If\ N \notin CS: S = U \cup V\ s.t. U \neq V, \{U,V\} \subseteq CS$$

Exit coalition

An agent from a coalition whose size is greater than one, i.e., not a singleton coalition, is randomly selected. The utility of the coalition minus the agent is calculated. If all agents in the remaining coalition improve their utility by removing the selected agent, the agent is removed from the coalition and forms a singleton coalition.

$$if\ \exists U \in CS\ s.t. |U| > 1: S = U \backslash \{i\},\ \ i \in U$$

Create a pair coalition

Two agents are randomly selected. The utility for the coalition containing just both agents is calculated. If the utility of both agents is improved in this new coalition, both agents exit the current coalition in favor of the new one.

$$S = \{i\} \cup \{j\},\ \ i \neq j, \{i,j\} \subseteq N$$

Create a trio coalition

Three agents are randomly selected. The utility for the coalition containing just the three agents is calculated. If the utility of all three agents is improved in this new coalition, all three agents exit the current coalition in favor of the new one.

$$S = \{i\} \cup \{j\} \cup \{k\},\ \ i \neq j \neq k, \{i, j, k\} \subseteq N$$

Defect coalition

A randomly chosen agent selects a coalition to which he does not belong. If joining this coalition improves his utility and the utilities of all members of the coalition, the agent defects from his current coalition and joins the new coalition.

$$S = \{i\} \cup U, i \in N, U \in CS \cup \emptyset$$

Split coalition

A coalition is randomly chosen, and a random subset of agents from the coalition are selected to form a separate coalition. If the members of the new coalition improve their utilities or the coalition that remained improve their utilities, the coalition splits into the two coalitions.

$$S_1 = X, S_2 = Y, X \cap Y = \emptyset, X \cup Y = U, U \in CS$$

Return to an individual coalition

An agent is randomly chosen. If that agent would be better off on their own, i.e., they prefer the singleton coalition to their current coalition; then, they leave their current coalition to form the singleton coalition. This is known as the individual rationality concept [84].

$$S = \{i\}, i \in N$$

Coalition Evaluation

Each of the seven suggestions is evaluated to determine if they are acceptable to members of the coalition (or either coalition is the case of a split). For each effect agent, their current utility is compared to the utility of the suggested coalition. If all the affected agents would experience an increase in utilities, then the suggested coalition forms. That is:

$$if\ \forall i \in S, u_i(S) > u_i(C_i)\ then\ C_i \coloneqq S, \forall i \in S$$

The utility that each player gets is the preference for the coalition in which they find themselves.

Coalition Updating

If a new coalitions form, then the agents of that coalition simply change their Coalition Membership number to a unique identification number assigned to the new coalitions. The forming of new coalitions will affect the utilities of many of the agents, but this information is updated when needed.

Using the ODD Protocol, I have described in detail the ABCG model and its functions for creating and changing coalition structures. The next section describes the experimentation I use to examine the ABCG model and determine its validity for the purpose of finding a coalition structure that is a member of the core solution of the defined hedonic game.

3.5 EXPERIMENTATION AND VALIDATION

“In the study of systems, the modeler focuses on three primary concerns: (1) the quantitative analysis of the systems; (2) the techniques for the system design, control, or use; and (3) the measurement or evaluation of the system performance" [85]. Experimentation with respect to simulation is generally the computer-based statistical sampling [86]. It represents one realization of the model but, due to the stochastic component, it requires multiple instances to perform statistical analyses on the system. Experimentation can also incorporate exploring differences in output performance, responses, that result from changes in input parameters and structural assumptions, factors [86]. In this section, I discuss the detailed steps in the experiments that I perform using the ABCG model to allow us to quantitatively analyze the model and evaluate the model's performance. I use these experiments to validate the model.

Validation is defined as building the right model [86, 87]. There are numerous techniques for validating models. Klügl [88] describes a validation framework that consist of four major techniques: face validation, sensitivity analysis, calibration, and statistical validation. Law [86] defines three processes: compare to an existing system, subject matter expert reviews, and sensitivity analysis. Face validation consists of experts assessing animation, outputs, and immersion – looking through the eyes of a single agent. This equates to Law's subject matter expert review. "Sensitivity analysis shows the effect of the different parameters and their values [88]." Primarily it reflects the effect of changing inputs on the system's outputs. Calibration is the setting of appropriate values for parameters within the model. Statistical validation is validation of the outputs of the system using generally accepted statistical methods.

I utilize three distinct experiments that allow me to explore the model and its relationship to its underlying theory and implement a use case to demonstrate its appropriateness in certain types of research. The structured design of experiments creates a plan for quantitative analysis of the model, model design, and measurement of the model outputs. In other words, the experiments are designed to allow me to validate the model. Figure 8 shows the experiment design structure for this research.

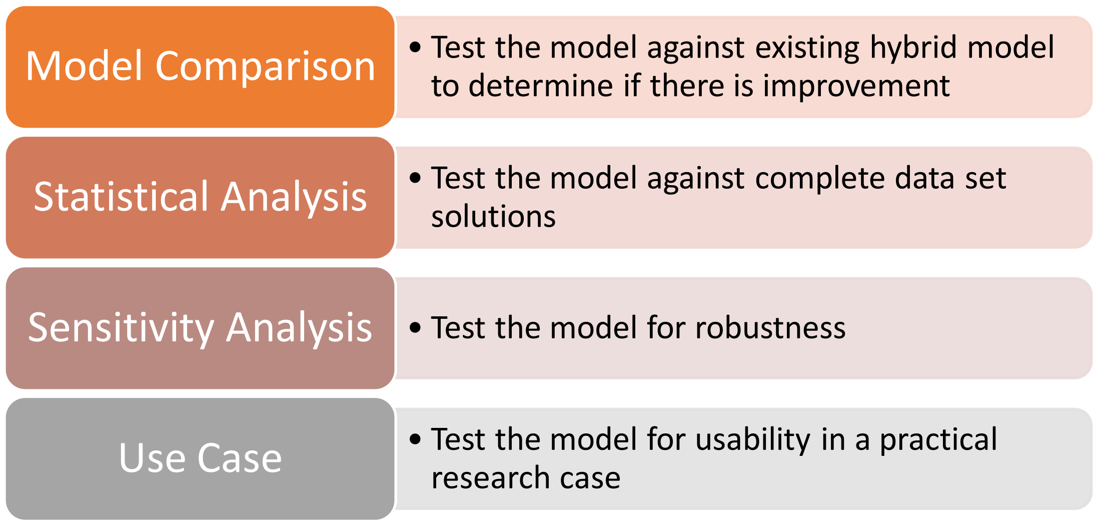

Model Comparison
• Test the model against existing hybrid model to determine if there is improvement
Statistical Analysis
• Test the model against complete data set solutions
Sensitivity Analysis
• Test the model for robustness
Use Case
• Test the model for usability in a practical research case

Figure 8: *Experiment Design Structure*

The first experiment is designed to test the ABCG model against an existing model. The experiment compares the results of the two models to determine if an improvement is made in the new model. It must be noted that a bias exists in the experiment. The original model was altered to fit the modified glove game. The original design included the limitation of a Moore neighborhood for selecting agents. The changes are my own designs, but the basic structure of the various methods remain faithful to the original model. The models are executed on the same set of data and the results are compared to the core solution. The second experiment is designed to test the effectiveness of finding a core member. The model is executed using the strict hedonic games for which the solution is known. It tests a wider range of agent sizes and number of games. The results of the model are then statistically validated against the solution. The third experiment allows us to perform sensitivity analysis on the model. In this experiment, I use the same data from the second experiment but alter the initial condition to determine its effect on model results. The model is designed to initialize to the grand coalition. This experiment determines if this initial condition has an impact on the end state of the model. In other words, does the initial configuration affect the outcome?

The fourth experiment is a use case of a common research problem. It utilizes hedonic glove games. In the first experiment, the glove game was simplified to only one preference and a limited number of agents; this experiment is closer to the simple market assignment game. Agents have different preferences and different levels of resources that allow them a novel means to identify their utilities for various coalitions. Unlike the strict hedonic games, ties are allowed. This broadens the possible values and creates widely different core solutions.

3.6 EXPERIMENT #1: MODEL COMPARISON

In this experiment I compare my ABCG model to the Collins-Frydenlund model. Axtell, Axelrod, Epstein and Cohen [89] bring to light the need to compare like models to create "alignment of computational models". They stress the fundamental need to determine if two models that claim to express the same phenomenon can produce the same results. I examine this by preparing both models to accept the same input and initial conditions and statistically comparing the output.

There are, however, several challenges and shortcomings to this experiment. First, the original model, while detailed in the interaction steps, incorporate a Moore's neighborhood in its selection process. This is removed for comparison purposes but fundamentally alters the model. The changes in the Collins-Frydenlund model are subject to interpretation by the research team; this potentially produces some bias. Finally, models are purpose built; altering the model requires re-validating it to its new purpose. Essentially, this version of the model I believe is sufficient and valid for the purpose of comparison. However, it should not be construed as having any effect or implication on the original work produced.

3.6.1 MODEL DESCRIPTION

I initialize the Collins-Frydenlund model in the same manner as the ABCG model utilizing a coalition value variable to express the individuals' preferences for the coalition with respect to all other coalitions of which they are a part. The interaction portion of the Collins-Frydenlund model consists of 10 steps listed below. I realize the steps in the NetLogo program in Appendix.

The Collins-Frydenlund model selects every agent in a random order and has them successively attempt to find a coalition that improves the value for all members of the new coalition. Once any change is selected, the model moves to the next agent.

3.6.2 PARAMETERS:

Time Steps: 25,000 ticks.

NetLogo simulation time runs at an arbitrary number of discrete steps defined as a "ticks" [79]. My model is designed such that within the space of each tick there is a full execution of the model sequence. The model sequence described above is executed complete at each tick. A sample number of runs was executed, and it was determined that, for the size of the games being used, steady state is reached using on average less than 10,000 ticks. The number of ticks is arbitrarily selected to ensure that number is exceeded. Since the number of ticks is consistent between the two models, this should not impact the comparison.

Number of Runs: 30 per game.

I use a Monte Carlo method of sampling and set the number of runs at 30 for each game. Mendenhall and Sincich [90] note that the size of the sample required is dependent upon the nature of the sampled population. Sampling populations can near a normal distribution with sample sizes as small as 25. I assume 30 to be sufficiently large; however, I recognize that the samples are not independent and identically distributed (IID), but the population has a finite mean and variance. Although I do not meet all the necessary criteria, I still infer properties of the central limit theorem: “If a random sample of n observations, $Y_1, Y_2, \dots, Y_n$, is drawn from a population with finite mean μ and variance σ^2, then when n is sufficiently large, the sampling distribution of the sample mean $\bar{Y}$ can be approximated by a normal density function [90].” As with any assumption, the risk of this is that the sample size is insufficient to accurate represent the population. However, due to the nature of the experiment, I believe the risks to be minor.

Number of Agents: minimum of 3; maximum of 9

The number of agents selected is intended to show the feasibility of the work. The minimum number of three agents allows for simple traceability while the maximum number of ten expands the experiment significantly enough to differentiate between the models.

Number of Games: 50 per game size.

Initial Condition: Grand Coalition.

The agents will initially be arranged in a single coalition.

3.6.3 DATA INPUTS AND OUTPUTS

Data inputs are files containing agents’ preferences and initial resources – number of left gloves and number of right gloves. The number of gloves is a randomly chosen number between zero and nine. Output is a coalition structure.

3.6.4 ANALYSIS CRITERIA

The analysis of this experiment will aid in determining if the ABCG model performs at least as well as the Collins-Frydenlund model with respect to the implemented theory. Both models will be compared to the known results. The measurement is the frequency with which the model is able to correctly match a core solution coalition structure. The comparison between models is the determined frequency for each agent-set size and the overall average frequency.

3.7 EXPERIMENT #2: STATISTICAL ANALYSIS

This experiment is designed to statistically evaluate of the ABCG model's ability to find a core solution member. Utilizing the randomly generated games I execute the model, retrieve a core solution, and compare that to the core solution set determined by the naïve algorithm.

3.7.1 PARAMETERS:

Time Steps: 50,000 ticks.

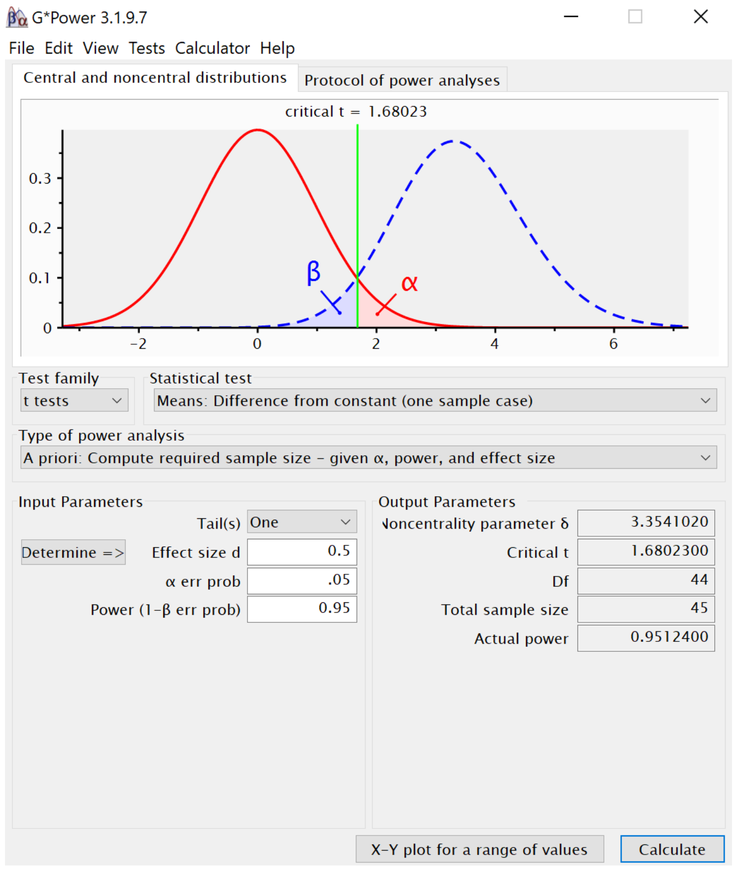
I execute the model for 50,000 ticks. The 50,000-tick mark is arbitrarily chosen. The intent is to ensure, as well as possible, that steady state has been reached. This does not guarantee that steady state will be reached. However, for additional reassurance, I sampled 5 games from each of the agent sizes (13 – 15) to review the impact of the increased number of possible coalitions and executed 10 runs each. I determined that steady state is generally reached in less than 30,300 ticks. There is still a risk that steady state is not reached within this timeframe.

Number of Runs: 30 per game.

As with Experiment #1, I use a Monte Carlo sampling with the intent that the sample size should be sufficient to approach normal.

Number of Agents: minimum of 4; maximum of 15.

My experiment is executed on games with 4 agents on the low end and 15 agents on the high end. The low end was to eliminate examining trivial games. For example, a 1 agent game only has 1 possible coalition structure and is therefore automatically the core solution. Similarly, a two-agent game only has two possibilities and is trivial to determine which option is best suited for the agents. My requirement on the high end was to select an agent set size that ensures the games are solvable in a reasonable amount of time. The designation of 15 agents is in line with the existing literature. For example, Collins et al. [75] use a maximum of 13 and Rahwan et al. [12] use a maximum of 25. The former solves for the core solution while the latter solves for maximum social welfare which requires fewer computations.


G*Power 3.1.9.7
File Edit View Tests Calculator Help
Central and noncentral distributions
Protocol of power analyses
0.3
0.2
0.1
0
−2
0
2
4
6
β
α
Determine =>
X–Y plot for a range of values
Calculate

critical t = 1.68023

Test family: t tests

Statistical test: Means: Difference from constant (one sample case)

Type of power analysis: A priori: Compute required sample size – given α, power, and effect size

Input Parameters

Tail(s)	One
Effect size d	0.5
α err prob	.05
Power (1−β err prob)	0.95

Output Parameters

Noncentrality parameter δ	3.3541020
Critical t	1.6802300
Df	44
Total sample size	45
Actual power	0.9512400

***Figure 9:** G*Power statistical power test*

Number of Games: 45.

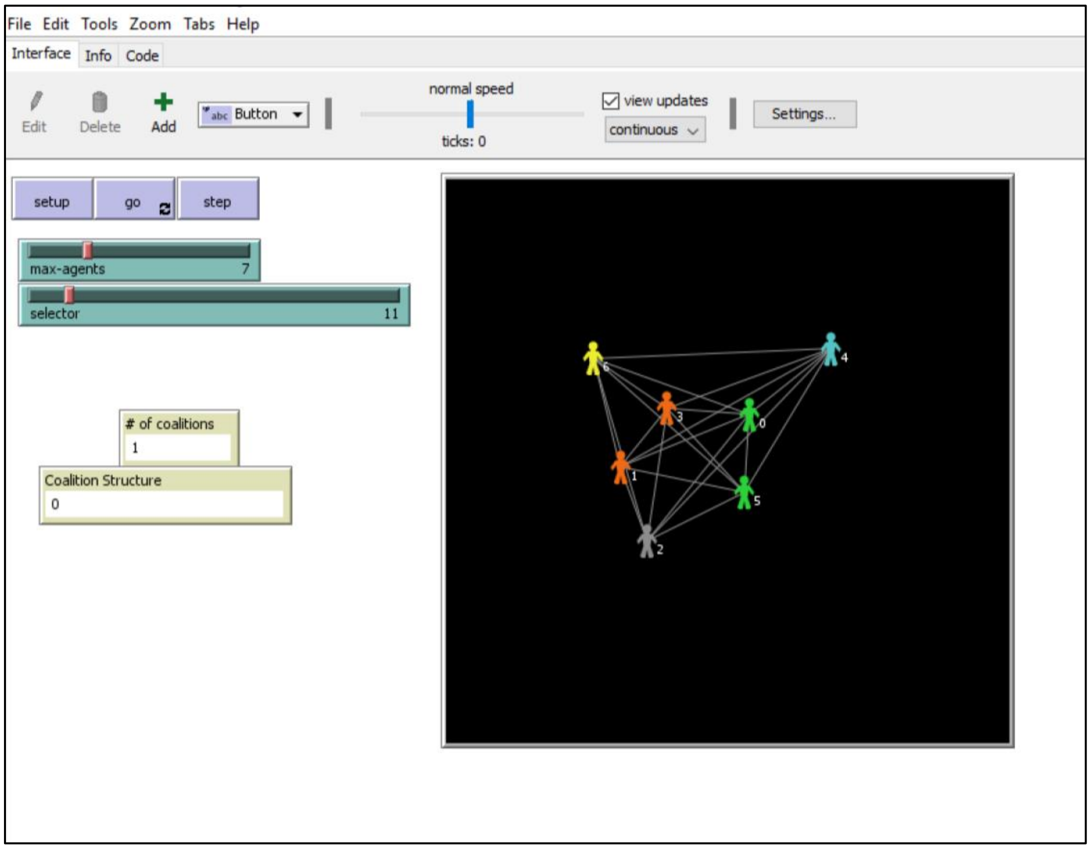
To ensure that I have adequate sample sizes, I perform a statistical power analysis. “The power of a statistical test of a null hypothesis (H_0) is the probability that the H_0 will be rejected when it is false, that is the probability of obtaining a statistically significant result” [91]. It is based on the mathematical relationship of the power, the population effect size, the significance level, and the sample size. There are several different software packages that aid in computing statistical power tests. I utilize G*Power [92]. G*Power is an open-source general power analysis program.

For the statistical power calculation, I desire a confidence level of $\alpha = 0.05$ and a power of $1 - \beta = 0.95$. Since I have no data to support the effect size, I assume a neutral value of 0.5. Figure 9 shows the calculation for the sample size. I execute 45 unique games per agent size group. That is, there are 45 unique 4-agent games; 45 unique 4-agent games; 45 unique 6-agent games, etc.

Initial Condition: grand coalition

The grand coalition is the coalition structure with only one coalition containing all members of the set. Sandholm et al. [30] demonstrates that the best case cannot be found unless there is a search of the grand coalition and all the possible paired coalitions. By starting with the grand coalition, I ensure that at least one of these conditions is met. The interaction routines attempt, but do not guarantee, that the second condition will be met through the model execution. The structure of my model only allows a coalition change if it is beneficial to all agents of the new coalition; therefore, if the grand coalition is a core member, the interaction routines will not change the coalition structure. However, if the grand coalition is not a core member, it is quickly eliminated from consideration.

Figure 10 displays the graphical user interface (GUI) for the ABCG model and demonstrates the initial configuration for a 7-agent game. As mentioned, the game is initialized with all agents in a single coalition represented by the links between agents. The max-agent slide bar allows for designation of the number of agents in the game. The selector allows for the selection of an existing game previously solved using the naïve algorithm. The number of coalitions tracks how many separate coalitions currently exist in the game and the coalition structure, once the game execution begins, will show which agents are in which coalitions.



max-agents 7

selector 11

# of coalitions 1

Coalition Structure 0

Figure 10: *Initial configuration in ABCG model*

3.7.2 DATA INPUTS

The data input consists of the coalition value array for each of agents in the game. The array contains the coalition which the agent is a member and the value of that coalition to the agent.

3.7.3 OUTPUTS

The output of the ABCG model is a coalition structure.

3.7.4 ANALYSIS CRITERIA

The experiment is designed to test the frequency with which the ABCG model correctly identifies a member of the core solution set for each game. The output coalition structure is compared to the various core members. The result of the comparison is a binary representing a match or no match. The primary statistical measurement is the percentage of times the model was able to correctly identify a core member. Additionally, I review the duration statistics for the ABCG model compared to the naïve algorithm. Since I know the naïve algorithm is NP-complete for large agent set sizes, this measurement allows us to make decisions as to the trade-off between execution time and precision of results.

The criterion for my analysis is ensuring that I my model finds a member of the core solution with an accuracy level of 95%. This level is measured against games with a non-empty core. It is important to realize that not all games have a non-empty core; however, I generally do not know this ahead of time. Collins, Etemadidavan, and Khallouli [75] show that the core results in an empty set an average of 5.27% for cooperative games with 3 to 13 agents, and 7.09% for 10 or more agents. My algorithm will never match to an empty core.

The model returns a coalition structure regardless of whether the core is empty or not. There are advantages and disadvantages to this. Researchers are always provided a result that is, at the very least, individually rational and generally more stable than previously tested coalition structures. However, without the knowledge of whether or not the core is empty, false assumptions can be made. As with all models, researchers need to consider the assumptions and constraints of the model carefully prior to application.

3.8 EXPERIMENT #3: SENSITIVITY ANALYSIS

This experiment is designed to test my assumption that beginning with the grand coalition provides the greatest opportunity for the model to resolve to a core member coalition structure. The previous experiment provides us the initial data for comparison. This experiment will first test the opposite initial condition – each agent is a member of the singleton group. That is, each agent is in a group of one and the number of coalitions in the coalition structure is equal to the number of agents. The second part of the experiment will test the effect of the initial condition being randomly chosen.

3.8.1 PARAMETERS:

The parameters are generally the same as the previous experiment with the exception of the initial condition.

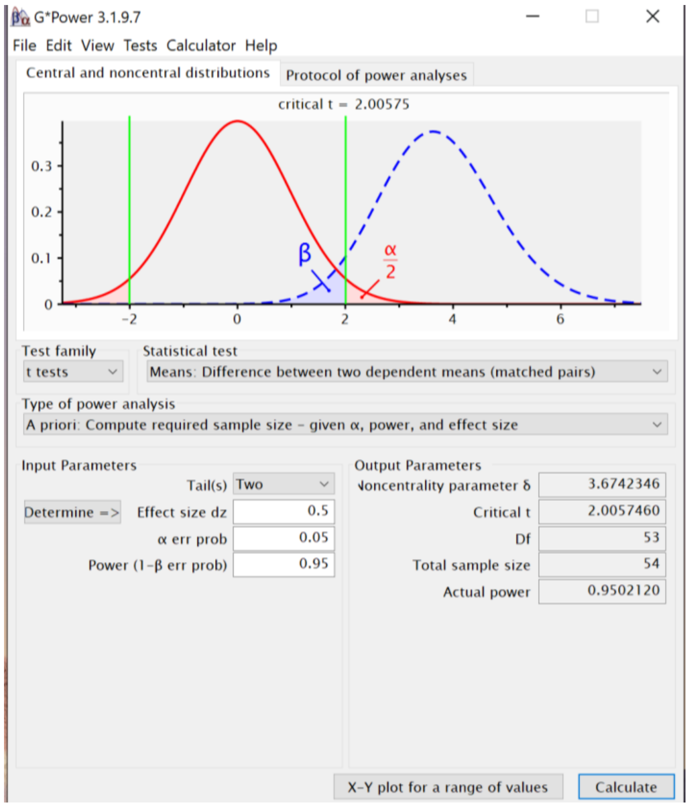
Time Steps: 50,000 ticks.

Number of Runs: 30 per game.

Number of Agents: minimum of 4; maximum of 15.

Number of Games: 54 per game size.

The statistical measure for this example differs from the previous. In Experiment #2, I compared frequencies to determine how well the model results matched the known core solution. In this test, I deem to determine the differences between matched pairs. To ensure that my sample size is adequate, I again employ the G*Power program. I alter the calculations to adjust for a two-tailed test. I determine that this experiment requires 54 samples to ensure the same statistical power used in Experiment #2. Figure 11 shows the results.


G*Power 3.1.9.7
File Edit View Tests Calculator Help
Central and noncentral distributions
Protocol of power analyses
critical t = 2.00575
0.3
0.2
0.1
0
−2
0
2
4
6
β
α/2
X-Y plot for a range of values
Calculate
Determine =>

Test family: t tests

Statistical test: Means: Difference between two dependent means (matched pairs)

Type of power analysis: A priori: Compute required sample size – given α, power, and effect size

Input Parameters

Tail(s)	Two
Effect size dz	0.5
α err prob	0.05
Power (1-β err prob)	0.95

Output Parameters

Noncentrality parameter δ	3.6742346
Critical t	2.0057460
Df	53
Total sample size	54
Actual power	0.9502120

Figure 11: *G*Power sample size results for two-tailed t-test with 0.95 statistical power*

Initial Condition: singleton coalitions and random coalitions.

The previous experiment held the initial condition to the grand coalition. In this experiment, I execute first with the initial condition of each agent being in a coalition of one (singleton coalition). Next, I re-run each of the games starting with varied coalition selection. Agents will be randomly grouped into coalitions. Figure 12a shows the initial condition of singleton coalitions; Figure 12b shows a random selection of coalitions.

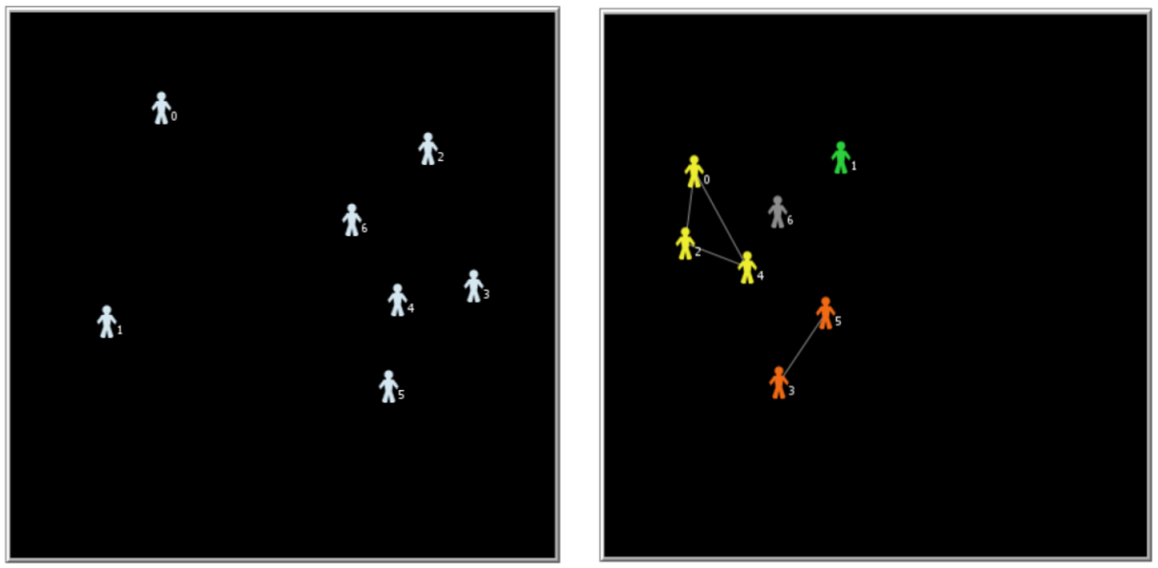


Figure 12: *(a) Initialized singleton coalitions in ABCG model; (b) initialized random coalitions in ABCG model*

3.8.2 DATA INPUTS AND OUTPUTS

Data inputs and outputs remain the same as the previous experiment.

3.8.3 ANALYSIS CRITERIA

The analysis of this experiment will aid in determining if the initial condition has any impact on the final coalition structure. While the previous experiment compared my results to the known solution, this experiment will compare the results of the three different initial conditions – the grand coalition, the singleton coalition, and the random coalition. I will attempt to determine if there are any significant statistical differences as a result of different initial conditions and, if a difference is found, which initial condition produces results that most frequently find a core member coalition structure.

For this analysis, I use a two-tailed test with the student's t-distribution to establish if the outcomes of the set of conditions varies from the outcomes of the second and third set of conditions. The data for this test is independent and identically distributed. I assume that the distribution is normal although there is a possibility that it is not. However, I consider this a minor risk as t-test are robust and flexible on this condition.

3.9 EXPERIMENT #4: USE CASE – GLOVE GAME

Coalitions are groups of agents that "decide to act together, as one unit, relative to the rest of the [agents] … however, this arrangement will continue only as long as each player finds it desirable to act this way" [93]. In other words, coalitions are agents that unite for a purpose that provides them a benefit. A simplified way to explore this is using a pure exchange economy example. A pure exchange economy is one in which there are no produces and agents have only their initial endowments. An example of a pure exchange economy that has been used in economic studies is the glove game. My use case will be a derivation of the glove game described in Shapley and Shubek [72]. The agents of the game are traders of gloves with a unique initial endowment of left gloves *(L)* and/or right gloves *(R)*. Each trader has a utility for either pairs of gloves, $u^1(L,R) = \min\{L,R\}$, or the leather associated with gloves, $u^2 = \frac{L+R}{2}$. Traders form a coalition structure to maximize their utility.

Economists have used the glove game to examine possible outcomes of endogenous coalition formations. The benefit of such a structure is that it is simplified but has an identifiable basis. It represents varying preferences and valuations based on those preferences. Unfortunately, like most coalition formation problems, using more than a few agents quickly makes the problem unwieldy. My use case demonstrates the ability of my model to aid researchers study this problem for a larger number of agents. The mechanics of the model are the same as used in the strict hedonic model; the difference being in the valuation of coalitions. The downside to this model, like the strict hedonic model, is that the result is singular – it does not produce the entire core solution set – and the solution is not guaranteed to be a member of the core especially as the core may be empty.

The glove game use case shows that my ABCG model can provide a means for researchers to examine the functions of a pure exchange economy for larger set sizes. It also demonstrates the ease with which the simulation can be changed for use with different coalition value sets. The variation between the use case and the strict hedonic game is only the determination of the agents' values for each coalition. Studying a pure exchange economy has helped researchers achieve insight into coalition behavior and has aided in the development of new theories. I believe that the use of simulation can further validate those theories.

3.9.1 MODEL DESCRIPTION

The agents are randomly provided with initial resources (left and right gloves) and a preference for pairs or leather. The difference between the games is the way in which I determine the value of the coalitions for each agent. Tables 7-10 are a representation of a 3-agent glove game. In this game, agents a, b, and c each have an initial set of resources (left and right gloves) and have a preference utility, u^1or u^2, that identifies the preference for pairs of gloves or leather respectively. Two coalition values are determined for each coalition: the maximum number of pairs that can be achieved by the coalition and the half the total number of gloves. The individual agent's value for the coalition in this game is designated by an equal sharing of the gloves. Agents that prefer pairs will receive the total number of pairs divided by the total number of agents in the coalition; agents with a preference for leather will receive the total number of gloves in the coalition divided by twice the number of agents.

Table 7: *3-agent glove game example -- initial resources and preference*

Agents' Initial Resources and Preference			
Agent	**Left Gloves**	**Right Gloves**	**Preference Utility**
a	2	3	u^1
b	4	5	u^1
c	3	0	u^2

Table 8: *3-agent glove game example -- coalition and agent values*

	Coalition Value		Agents' Value in Coalition		
Possible Coalitions	u^1	u^2	**a**	**b**	**c**
(a)	2	2.5	2	--	--
(b)	4	4.5	--	4	--
(c)	0	1.5	--	--	1.5
(a, b)	6	7	3.5	3.5	--
(a, c)	3	4	1.5	--	2
(b, c)	5	6	--	2.5	3
(a, b, c)	8	8.5	2.67	2.67	2.83

Table 9: *3-agent glove game example -- agent values in coalition structure*

	Agents' Values		
Coalition Structures	**a**	**b**	**c**
[(a) (b) (c)]	2	4	1.5
[(a,b) (c)]	3.5	3.5	1.5
[(a,c) (b)]	1.5	4	2
[(b,c) (a)]	2	2.5	3
[(a,b,c)]	2.67	2.67	2.83

Table 10: *3-agent glove game example -- core solution set*

Core Solution Set
[(a) (b) (c)]

Examining the possible coalitions shows that *agent b* would never form a coalition with any other agent as joining a coalition with any other agent decreases his utility. Therefore, only *agent a* and *agent c* are left to possibly combine. If *agent a* were to combine with *agent c*, the resulting coalition would be less for *agent a* than remaining independent – individual rationality.

Therefore, the only coalition structure that is not dominated and therefore a member of the core solution set is [(a) (b) (c)].

I employ the same techniques for testing my use case as I did for the strict hedonic game except for the coalition valuation step. For the strict hedonic games, I randomly ordered the agents' coalition preferences and valued the coalition accordingly. In the glove games, the coalition values for each agent are based on the predetermined division of resources. Within the ABCG model, as the coalitions are formed, the values for each agent in the coalition are calculated and the coalition is tested to determine whether the new values benefit all agents in the coalition.

3.9.2 PARAMETERS:

The parameters are generally the same as the first experiment.

Time Steps: 50,000 ticks.

Number of Runs: 30 per game.

Number of Agents: minimum of 4; maximum of 15.

Number of Games: 20 per game size.

Initial Condition: Grand Coalition.

3.9.3 DATA INPUTS AND OUTPUTS

I utilize the basic structure of the naïve algorithm presented earlier as my starting point for creating the set of games. This version of the algorithm is altered to provide agents with a random number of left gloves, right gloves, and an indicator for a preference of leather or pairs instead of a preference value for each coalition. The coalition preference value is calculated for each agent in each possible coalition using the formulas in the model description section above. Once all the coalition values are computed, the program solves for the core solution.

The sample size is 20 games per agent set size. Sampling shows that this version of the glove game produces an empty core about 75% of the time. Sampling at 600 per agent-set size (20 unique games executed 30 times) should not degrade my statistical power. The output for each run of the ABCG model is a coalition structure.

3.9.4 ANALYSIS CRITERIA

The criteria for this analysis is the same as Experiment #1. The experiment will test the frequency with which the ABCG model correctly identifies a member of the core solution set for each game and determine if the output coalition structure is a member of the core. The analysis is designed to measure if I achieve find the core at least 90%.

3.10 SUMMARY

In this chapter I provided the research details for creating and validating my ABCG model. I began by describing the methods used to create hedonic games and algorithmically solve those games. I utilized a naïve algorithm which solves the hedonic games in a "brute-force" type effort. While this method provides a complete solution, it is computationally intensive and is impractical for large agent set sizes. However, it provided a significant test set for validation of the ABCG model.

I described the heuristic model I designed to find a member of the core solution set. The model is an Agent-based model that incorporates cooperative game theory into its computational structure. It utilizes seven interaction routines to create different coalitions. Each new coalition is evaluated using the concepts that determine the core solution. Coalitions that improve the coalition value for every member of the coalition are incorporated into a new coalition structure. This continues for a pre-determined amount of simulation time known as "ticks". The final coalition structure is then compared to the core solution of that game to determine if the model was able to find a core member.

The ABCG model does not explore the entire solution space as the naïve algorithm does. Instead, it uses stochastic sampling to create various coalitions. Since I introduce random sampling, statistical analyses are used to interpret the results. I utilized G*Power to determine the required sample size for a 95% statistical power. I also sampled to determine the appropriate number of simulation ticks required to reach a “steady state”, but intentionally went beyond that number to potentially address any outliers.

Statistical analysis of results is considered the most important form of validation [86]; however, I further validated my model using sensitivity analysis. The sensitivity analysis examines the impact of initial conditions on the model. My first set of experiments was conducted with an initial condition that placed all agents in one coalition – grand coalition. The assumption was that by starting with the agents together, I was more likely to find a core member or at least ensure that I have tested Sandholm et al.’s [30] lowest level. However, there is no evidence that this initial condition is either optimal or impactful. I therefore designed an experiment to test the impact of the initial configuration on the ability to find a core member. Like the first experiment, the stochasticity of the model required us to generate several samples and statistically analyze these to determine at a 95% accuracy level if the initial condition has an impact.

I establish my validation techniques and my null hypotheses for testing. Finally, I show through a use case the value that my model adds to the research community. Various experiments were conducted to test the statistical validity of the model, the improvement of the model over existing models, and the structural validity of the model. Senge and Forrester [94] discuss the need to validate the model structure through the generated behavior: “Tests of model behavior evaluate the adequacy of model structure through analysis of behavior generated by the structure." I examine the impacts of my model structure by testing variations on the order in which routines are executed. This allows me to determine if the generated behavior is truly the result of the base assumptions or is artificially induced by the routine ordering.

Table 11 provides a listing of each of the experiments performed along with their corresponding descriptions and objectives.

Table 11: Set of experiments used to validate and determine the viability of the ABCG model in finding a core member coalition structure.

Experiment #	Experiment Name	Description	Objective
1	Model Comparison	This experiment is designed to compare the ability of the ABCG model to find a core member to the Collins-Frydenlund model ability to find a core member.	Determine if the ABCG model is at least as capable of finding a core solution member than the previous Collins-Frydenlund model.
2	Statistical Analysis	This experiment is designed to test the statistical validity of the ABCG model using a sample set of strict hedonic games and testing against a null hypothesis.	Determine if the number of times the algorithm finds a core solution member is statistically significant.
3	Sensitivity Analysis	This experiment is designed to test the impact of the initial condition of the coalition structure on the ability to find a core member	Determine if there is a statistically significant difference between the initial condition of the grand coalition compared to an initial condition of all singleton groups or random coalitions
4	Glove Game Use Case	This experiment is designed to show the ability of the ABCG model to be used in conjunction with existing research techniques and problems.	Determine if the ABCG model can be effective in finding a core model for an existing research problem

This chapter described the research method and steps used to determine whether the ABCG model provides a reasonable means to find a core member. The next chapter discusses the results and analysis of this method.

4.0 RESULTS AND ANALYSIS

In this chapter I provide an analysis of my model experimentation. There are four distinct experiments set out in the method chapter that my study addresses. In the first experiment, I demonstrate that my model provides an improvement over a model currently found in the literature. I provide a direct comparison between the ability of my model to find a core member and the Collins-Frydenlund model's ability to find a core member. In the second experiment, I show that I can increase the number of agents in the coalition structure solve games by directly assigning the coalition values. I also show examples of the difference in algorithm execution time between the naïve algorithm and the ABCG model. Experiment #3 provides insight into the impact of the model's initial state. My experiment repeats the glove game from Experiment #1 using the larger coalition sizes to show the model's usefulness in a common research example.

These four experiments exhibit the usefulness of the model in studying strategic coalition formation and aid in answering the research question: How can an ABM algorithm be designed such that its outcome is a member of the core solution set of a hedonic game greater than 90% of the time and the computation of the outcome completes in polynomial time? The frequency experiment provides an empirical comparison to show that the model can find a member of the core solution set greater than 90% of the time. The duration measurements demonstrate that the model runs in polynomial time despite the size of the coalition structure. I also examine the sensitivity of the model to the initial coalition structure to ensure that the model can be run with any starting coalition structure and still be expected to perform in the same way.

The remainder of this chapter is organized in three major sections. The first section describes the games that are used in the experiment. It provides a description of the data characteristics. The second section details the experiments and the results gleamed from each of the experiments. The experiments section is broken down into four subsections: one for each experiment. The final section summarizes the analysis of the experiments and draws conclusions based on the results.

4.1 GAME DESCRIPTIONS

As previously mentioned, I generate the input data by randomly assigning coalition values to each agent in the strict hedonic game and randomly assigning resources to each agent in the glove game. Experiment #1 used a small data set to determine if my model provided any significant improvements over a similar model found in the literature. The data set consisted of agent-set sizes ranging from 3 agents to 9 agents. Each agent set size consisted of 10 unique games for 80 unique games. For experiments 2 and 3, I executed the naïve algorithm 600 times to create unique strict hedonic games – 50 unique games for each of the 12 agent-set sizes 4 through 15 – and 600 unique hedonic glove games. Out of those strict hedonic games, 536 of the 600 games or 89.33% of the games had a non-empty core. Figure 13 shows the breakdown by agent-set size for non-empty core games versus empty-core games. Each of the agent-set sizes had at least 40 games with a non-empty core. Although this is slightly less than the objective of 45 games each, I considered the overall sampling to be sufficient.

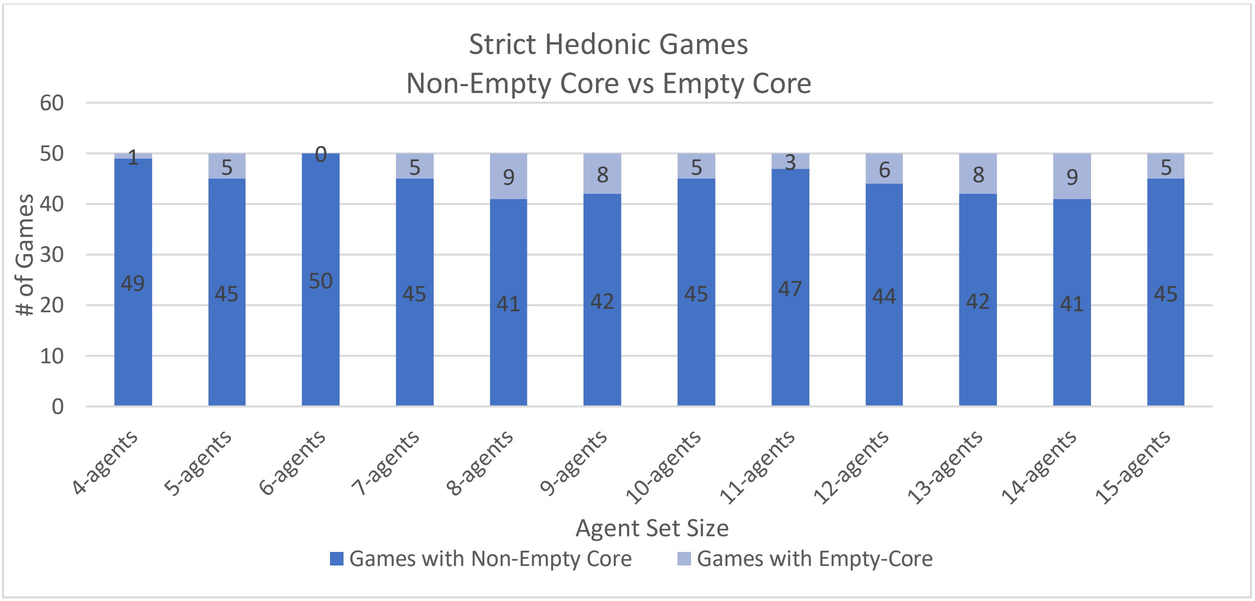


Figure 13: *Strict hedonic game breakdown between non-empty core games and empty core games*

The hedonic glove game experiment yielded greater variability in the core solution. Unlike the strict hedonic game, there are ties allowed for an agent's preference. Under this condition, the core solution sets vary from a significant number of games having an empty core to games with thousands of core solutions. In the first execution of the naïve algorithm for the hedonic glove game, only 141 out of the 600 games, 23.50% had a non-empty core. Figure 12 shows the breakdown among the various agent-set sizes. This set does not provide a sufficient non-empty sample set to achieve the desired statistical power. Therefore, I expanded the number of games to 100 per agent-set size. Figure 13 shows the expanded set.

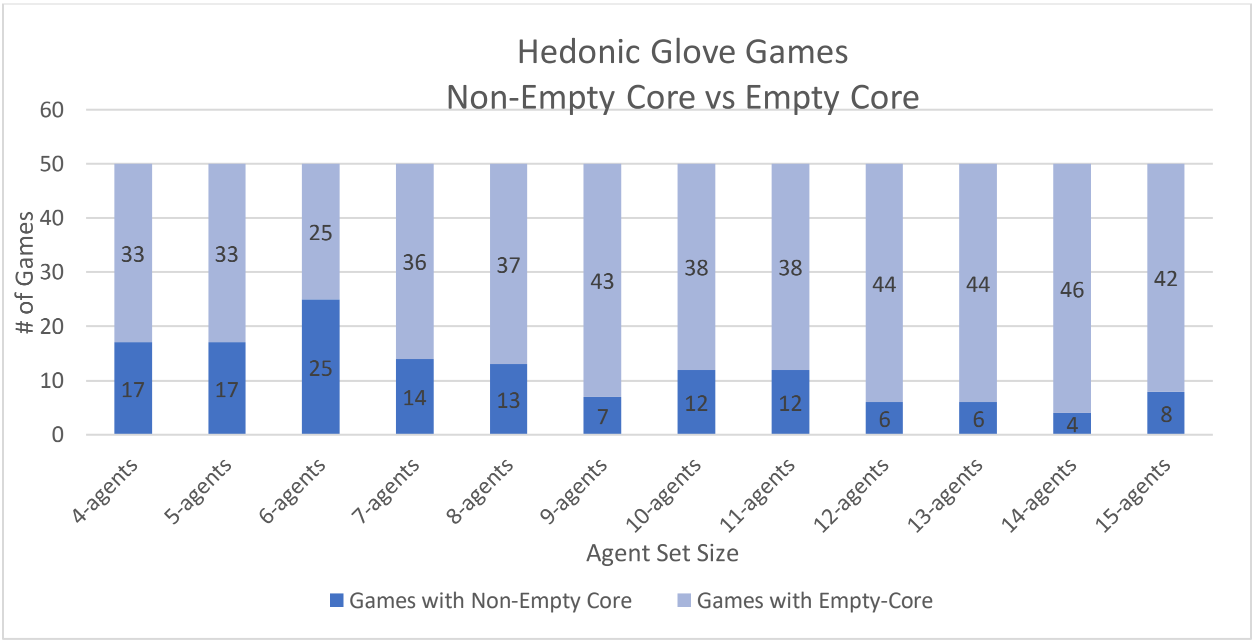


Figure 14: *Glove game breakdown between non-empty core games and empty core games*

The next section reviews the model experiment and their results.

4.2 ABCG EXPERIMENTS

This section reviews the results of the four experiments listed in the previous chapter. The first experiment compares my model to the Collins-Frydenlund model. My effort with this experiment is to determine if my model improved the current state found in the literature. The second experiment is designed to test the model in a broader setting; that is, using strict hedonic games with larger sizes. The third experiment provides a sensitivity analysis for the model. It tests the effects of the initial condition. Finally, the fourth experiment provides a use case for the model. It attempts to find a stable core solution in a known theoretical economics problem, the glove game, to demonstrate its usefulness in examining hedonic games with several agents.

Experiment #1 results chart the frequency with which the Collins-Frydenlund model and the ABCG model were able to identify a core member for each Glove Game. The experiment shows the improvements achieved with the new algorithm but also indicates areas for potential improvements. Experiment #2 changes the parameters from the glove game to a more generic hedonic game with up to 15 agents. Again, the frequency with which a core member is identified is graphed. In this experiment I see a wider range of results, partially due to games with an empty core. I will review the impact of empty-core games on the results as well as the benefits and drawbacks of the ABCG model results for empty core games. Experiment #3 results graph the differences in frequency based on different initial coalition structures. The model's initial conditions range from the grand coalition, i.e., all agents in a single coalition, to each agent in a coalition by itself, to randomly assigning agents to a coalition. The results indicate that there is no significant difference in model results due to the initial coalition structure. Finally, experiment #4 demonstrates the use of the model in a theoretical economic research problem. The problem is a simple market economy game. Many versions of the game exist. I use a modified version of the glove game assignment market. The results are then compared to the analytical results of a naïve algorithm for the glove game.

Each experiment is designed to demonstrate purpose and validity of the model. The experiments cover empirical validity, sensitivity analyses, and model comparisons. The next sections of this chapter provide the results and analysis of each experiment.

4.2.1 EXPERIMENT #1

Axtell et al. [89] emphasize the need for Agent-based modelers to be able to compare models for validation. The initial experiment is designed to compare the Collins-Frydenlund model to my ABCG model utilizing a modified Glove Game. I utilized a small data set to determine the accuracy with which the Collins-Frydenlund model achieves finding a core member then to examine improvements gained by using the ABCG model. The data set is generated using the naïve algorithm. This section discusses the results of the simulation runs.

The results focus on the comparison of the effectiveness of the two algorithms (the original Collins-Frydenlund algorithm and the new one presented in this paper). Figure 15 shows the percentage of times the original Collins-Frydenlund algorithm reached a coalition structure that is a member of the core solution set (i.e., a core partition).

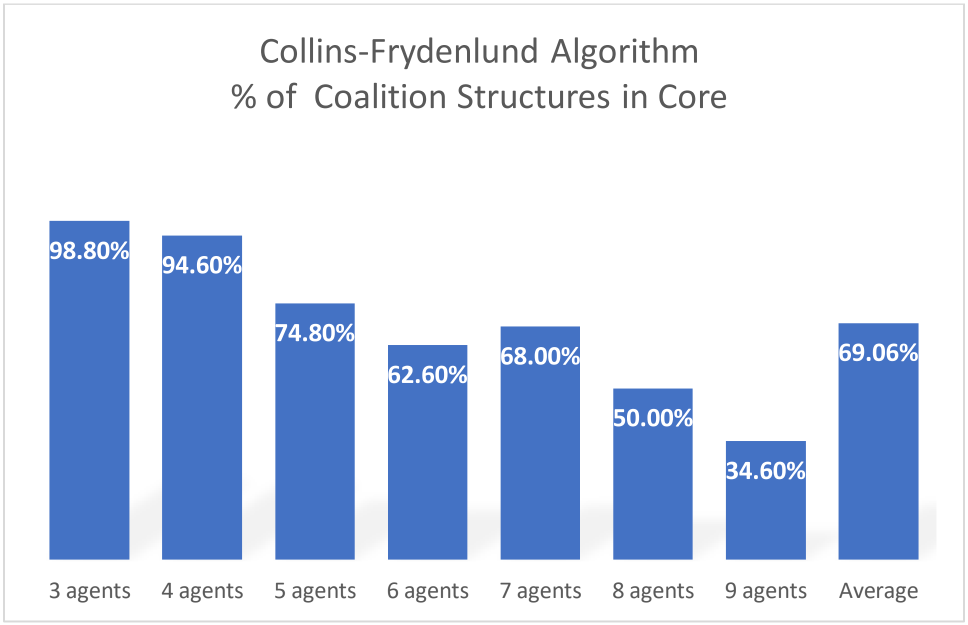


***Figure 15**: Percentage of times the converted Collins-Frydenlund algorithm simulation coalition structure was part of the core solution*

Figure 16 shows the results of my algorithm. The new algorithm significantly outperforms the old algorithm. Also, the new algorithm has a high rate of finding a core member as its solution. Overall, 96.1% of the games resulted in finding a core member under the new algorithm; that is, out of the 3500 games executed, the resulting coalition structure was a member of the core 3363 times. Games with three-agents, four-agents, and seven-agents always found a core member; 488 out of 500 five-agent games found a core member; 498 of the six-agent games resulted in a core coalition structure; and 499 games played with nine-agents achieved a core result.

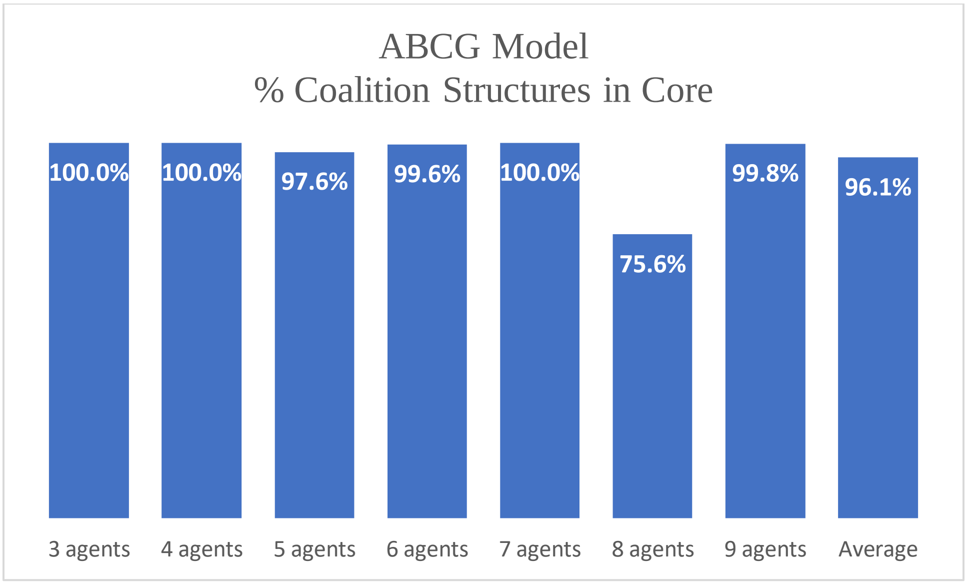


Figure 16*: Percentage of times the simulation coalition structure was part of the core solution.*

To verify that the result is statistically significant, I perform a one tail paired t-test with $\alpha = 0.05$ to determine if the mean difference in the sample population is significant. The null hypothesis is that the Collins-Frydenlund model is at least as good as the ABCG model at finding a core member. The alternate hypothesis is that the ABCG model is better. Table 12 are the results of the one-tailed t-test. The results indicate that I should reject the null hypothesis and accept that the ABCG model shows statistical improvements.

Table 12: *One-tailed test results for paired t-test between Collins-Frydenlund model and ABCG model*

	Collins-Frydenlund Model	*ABCG Model*
Mean	0.691	0.961
Variance	0.045	0.007
Pearson Correlation	0.369	
Hypothesized Mean Difference	0.000	
t Stat	-3.867	
P(T<=t) one-tail	0.003	
t Critical one-tail	1.895	

The only concerning result from this experiment is the eight agent games. The eight-player games, by contrast, significantly underperformed. Almost 25% of the time (122 out of 500), the eight-agent game failed to conclude with a coalition structure that could not be dominated. The details of these games are shown in Table 11. An examination of the games that failed to reach a core solution showed that these games reached a local maximum. The algorithm used for the ABM belongs to a group known as "greedy algorithms". These algorithms are based on achieving local optimal decisions with the hope of a global solution. However, the global solution is not actually represented or considered. Therefore, it is possible for a coalition to reach a value in which it is unwilling to reconsider given the rules for a change. That is the game stalls at a local maximum. Table 13, and resultant discussion, attempts to explain why this occurred with an example.

Table 13: *Agent coalition preference values for 8-agent game that does not achieve a core member solution; the first agent is represented by zero which is common in computing terminology. (a) Agent resources of the game; (b) Resulting coalition structures.*

	Agents' Resources							
Agents Resources	**Agent 0**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Agent 6**	**Agent 7**
Left Gloves	0	2	9	0	6	2	3	7
Right Gloves	4	4	4	8	3	1	5	9

(a)

	Agent Coalition Preference Values							
Core Coalitions	**Agent 0**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Agent 6**	**Agent 7**
(0, 5) (1) (2, 3, 4) (6) (7)	1	2	5	5	5	1	3	7
(0) (1) (2, 3, 4) (5) (6) (7)	0	2	5	5	5	1	3	7
(0) (1,5) (2, 3, 4) (6) (7)	0	2	5	5	5	1	3	7
Coalition Structure Achieved								
(0) (1,4) (2, 6) (3) (5) (7)	0	3.5	4.5	0	3.5	1	4.5	7

(b)

The first part (a) shows the initial resources for each of the eight agents. The core coalition section of the second part (b) is the set of coalition structures and their utility values within the core. Obviously, each player would like to obtain the highest utility. Part (b) also contains an example of a “sub-optimal” coalition structure that was achieved by the algorithm. To understand why a core partition was never achieved, I need to consider the linchpin coalition. The linchpin coalition in the core, in the game under consideration, consists of agents two, three, and four. A linchpin coalition is a coalition that occurs in all core partitions. Technically, the singleton coalition of agent seven is also a linchpin coalition, so is agent six’s singleton group. Thus, for a core partition to evolve, under my algorithm, from a given coalition structure, the linchpin coalition must be able to form. Unfortunately, in the example given above, this is not possible from the coalition structure achieved because of the limitation of the algorithm.

The coalition consisting of agents two, three, and four represents the highest utility value that those agents could achieve, and it would be expected that they would always be together. However, during the simulation, the coalition structure forms a coalition consisting of agents one and four prior to achieving the two, three, four combinations. As a group, neither agent one nor four improves their position by isolating themselves and agents one or four do not improve their position by joining any core member coalition structure. Therefore, agents one or four are not willing to change coalitions. Also, the coalition does not benefit from merging with any other existing coalition. Hence, under my algorithm, there is no possibility that the coalition, of player one and four, will change. In fact, there is no incentive for any coalition in the coalition structure to change, given the rules of the algorithm. This leads to a local maximum being reached.

The local maximum is not a member of the core and purely an artifact of the algorithm. This is due to the pair-wise nature of the algorithm; that is, at most, only two agents or coalitions are considered at any one time. For the linchpin coalition to form from the local maximum, the three agents, two, three and four, would need to be considered at the same time, within the algorithm step, which is not possible because they are in three different coalitions. My heuristic algorithm is, in this respect, limited. However, the results demonstrate that, within a margin of error, it is possible to arrive at highly probable stable coalition structures which may or may not be in the core. This problem of being trapped in a local maximum could be overcome by using approaches likes simulated annealing [95]. I will examine in later experiments other possible ways to address this issue.

These results demonstrate that the ABM is effective in achieving a stable coalition structure in which the players have no incentive to move to another coalition (at least under the algorithm restrictions). However, it does not identify every member of the core set; that is, not every possible core partition was reached by the algorithm. The ABM results tended to be biased towards one core partition over all others in the core. Figure 17 shows that only 36% of the possible core partitions were achieved. Further, there is a significant decrease in the average percentage of possible coalition structures reached when the number of players increases from 6 to 7. This is most likely due to the size of the core increasing with little to no change in the number of unique coalition structures achieved. Specifically, for the 10 games with 6 players, there are a total of 61 coalition structures in all their cores; this increases to 174 when there are 7 players. However, the number of unique coalition structures reached in the simulation is 15 and 16 for 6 and 7 players respectively. This is most likely due to the structure of the model and the order in which the combinations are put forth. Once certain combinations have been achieved, coalitions that are not blocked, there is no chance of altering them. While not a flaw, it is important to note that it occurs.

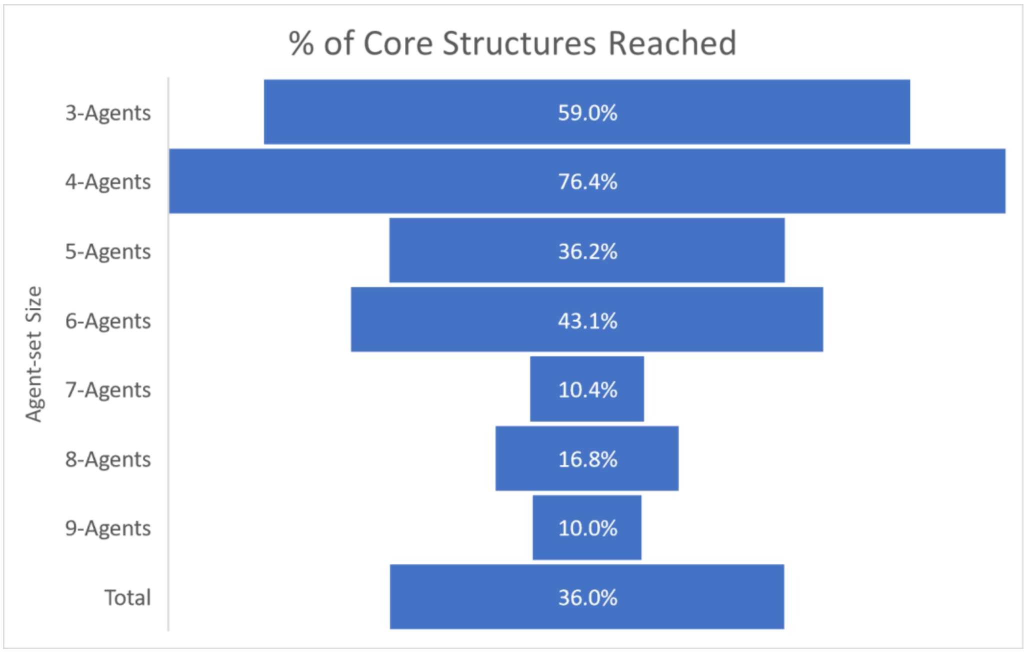


Figure 17: *Percentage of different core structures achieved using ABCG model*

4.1.2 EXPERIMENT #2

Experiment #2 is designed to test the algorithm in a more generic set of hedonic games and with a larger number of agents in the game. Game sizes range from 4-agent games to 15-agent games. The games I use are strict hedonic games; that is, ones in which each agent ranks each coalition by preference from lowest to highest, one being the least preferred coalition. Ties between coalitions are not permitted. In this experiment, I also update the ABCG model to address the issue of local maximums that was seen in the first experiment and to expand the number of coalitions tested at each simulation tick.

Figure 18 shows the percentage of runs in which a core member was properly identified. The results range from a low of 71.6% to a perfect 100% match with the average being the model finding a match 83.6% of the time. However, for several of these games, no core solution exists. One feature of my algorithm is that it returns a coalition structure regardless of whether the core is empty.

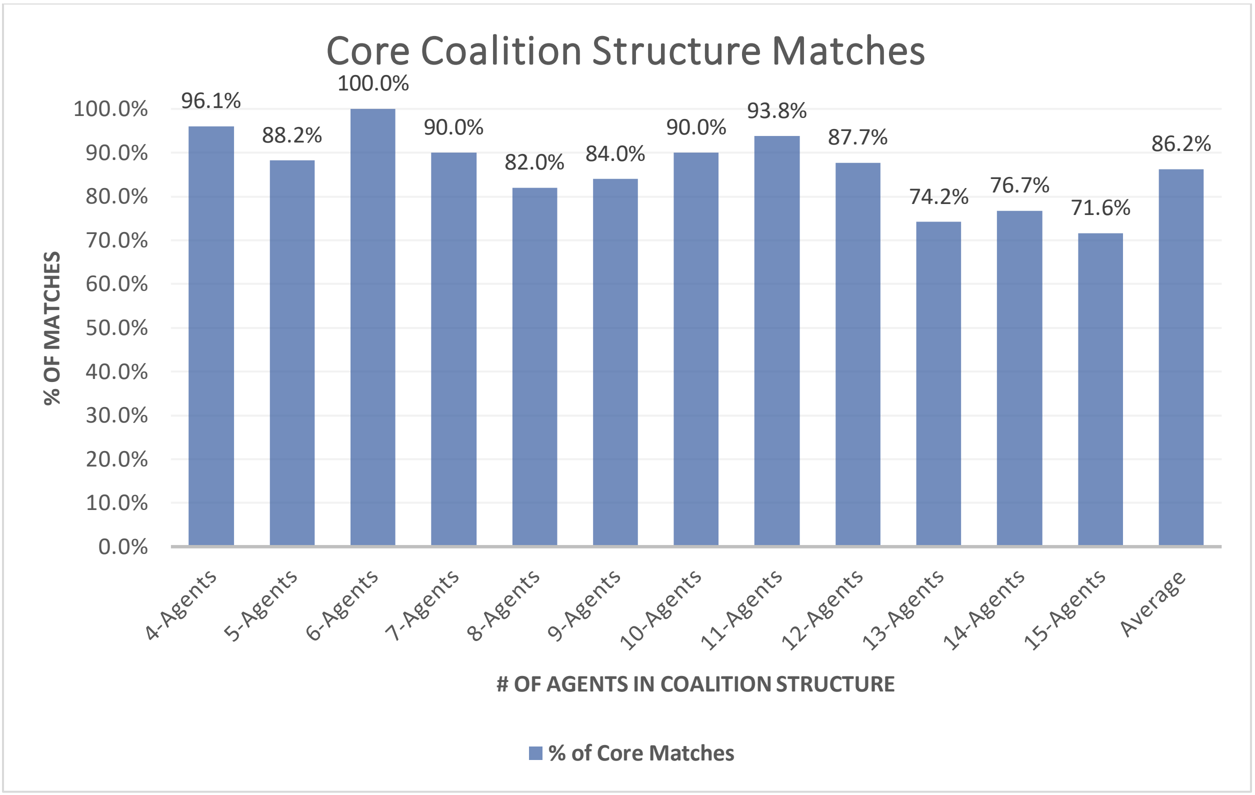


Figure 18: *Percentage of times the ABCG model result matched a core member*

In Figure 13 of the previous section in, I noted that overall, 10.67% of the games had an empty core. Therefore, the maximum potential match on average is 89.33%; I achieved 86.19% coalition matches. The ABCG model always returns a coalition structure. Games with an empty core, therefore, will never match the algorithm result. Figure 19 shows the frequency with which a core member was identified for all games with a non-empty core. A review of this chart shows that the model performs extremely well for games with 12 agents or less; however, there is a noticeable decline in the results for games with greater than 12 agents. I take a closer look at these games.

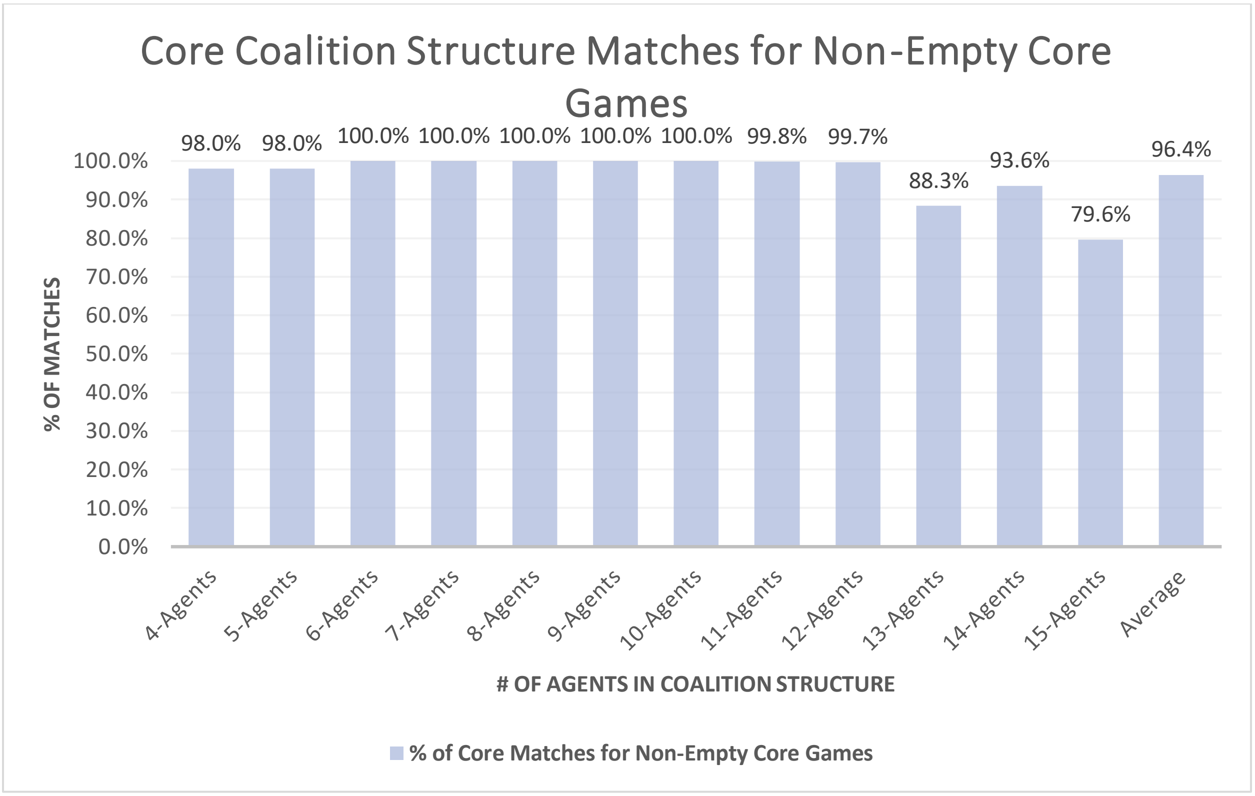


Figure 19: *Percentage of times the ABCG model was able to identify a core member for games with a non-empty core*

There are 50 unique games for each agent set size and each game was executed 30 times. The first series that shows a significant decline in properly identifying a core member is the 13-agent set of games. The hedonic games for 13 agents have 42 games with non-empty core and 8 games with no core solutions. Figure 20 shows the results for the 42 games with a non-empty core. The ABCG model was able to accurate identify a core coalition structure without a miss for 32 out of the 42 games (76.2%) and with greater than 95% accuracy for 81.0% of the time. However, in rare occasions, 6 times out of 42 or 14.3%, the model was less than 50% accurate in determining a core member. This implies that when the model is generally capable of finding a solution. However, there are games for which the heuristic is not well suited.

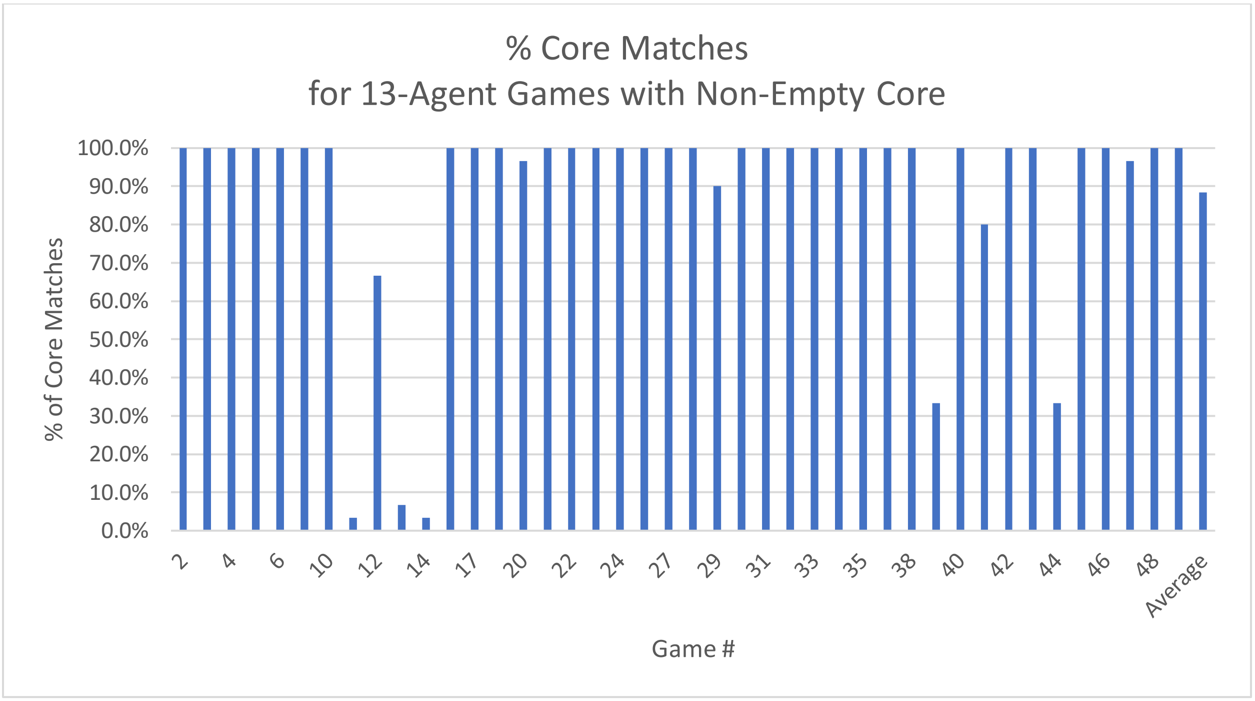


Figure 20: *Breakdown of 13-agent non-empty core game matches*

I examine game 11 in greater detail to try to understand why the algorithm is not well suited to this game. Game 11 in the 13-agent set has a success rate of 3.3%; a correct coalition structure was found only once in 30 attempts. The core contained three coalition structures:

- Core Coalition Structure 1 – [(0, 11) (1, 3, 5, 12) (2, 7, 8) (4, 9, 10) (6)]
- Core Coalition Structure 2 – [(0, 11) (1, 3, 5, 12) (2, 8) (4, 9, 10) (6, 7)]
- Core Coalition Structure 3 – [(0, 11) (1, 6) (2, 8) (3, 5, 7, 9, 12) (4, 10)]

The execution of the 30 runs of game 11in the ABCG model netted 4 unique coalition structures:

- ABCG Coalition Structure 1 – [(0, 11) (1) (2, 3, 6, 7) (4, 10) (5, 9) (8, 12)]
- ABCG Coalition Structure 2 – [(0, 11) (1, 4) (2, 3, 6, 7) (5, 9) (8, 12) (10)]
- ABCG Coalition Structure 3 – [(0, 11) (1, 10) (2, 3, 6, 7) (4) (5, 9) (8, 12)]
- ABCG Coalition Structure 4 – [(0, 11) (1, 3, 5, 12) (2, 7, 8) (4, 9, 10) (6)]

Table 14 contains the value each agent has for the core coalitions and the ABCG coalitions. The highlighted column on ABCG Coalition Structure Value 4 indicates that it is the only coalition structure that is a member of the core. Agents 0 and 11 consistently unite to form a coalition in both the core and the ABCG model. Agent 3's values indicate a potential local maximum concern. Once Agent 3 became part of the coalition (2, 3, 6, 7) there was never another coalition tested that improved its result. Therefore, no other coalition would be accepted by Agent 3.

Table 14: *Core and ABCG model coalition values for 13-Agent Game 11 analysis*

Game 11 Coalition Values	Core Coalition Structure Value 1	Core Coalition Structure Value 2	Core Coalition Structure Value 3	ABCG Coalition Structure Value 1	ABCG Coalition Structure Value 2	ABCG Coalition Structure Value 3	ABCG Coalition Structure Value 4
Agent 0	3,976	3,976	3,976	3,976	3,976	3,976	3,976
Agent 1	3,290	3,290	2,435	13	163	2,445	3,290
Agent 2	3,909	3,891	3,891	3,422	3,422	3,422	3,909
Agent 3	3,238	3,238	2,664	4,031	4,031	4,031	3,238
Agent 4	3,723	3,723	1,486	1,486	105	839	3,723
Agent 5	3,834	3,834	3,605	3,215	3,215	3,215	3,834
Agent 6	1,048	1,450	2,002	3,696	3,696	3,696	1,048
Agent 7	4,025	2,874	3,620	4,031	4,031	4,031	4,025
Agent 8	2,998	3,743	3,743	4,009	4,009	4,009	2,998
Agent 9	1,925	1,925	3,595	3,912	3,912	3,912	1,925
Agent 10	3,642	3,642	4,080	4,080	1,466	4,080	3,642
Agent 11	3,799	3,799	3,799	3,799	3,799	3,799	3,799
Agent 12	3,269	3,269	4,062	2,897	2,897	2,897	3,269

This concept is central to the function of Agent-based modeling. ABM is a "bottom-up" system; the coalition structures are determined based on the agents' rules and local interactions. There is no central authority controlling the agents' actions or examining the details of coalition structure. Instead, the agents choose to stay in an existing coalition or migrate to another coalition based on their preferences between the coalitions. The benefit in this type of example is the efficiency with which comparisons are made; the drawback is that not all structures are analyzed independently.

While I have attempted to mitigate the local maximization through increasing the variations and total number of combinations tested, I have not eliminated it. An alternative method for mitigating local maximization is simulated annealing. Simulated annealing is a technique that has been used as a metaheuristic for combinatorial optimization problems [96]. It is “a local search algorithm (meta-heuristic) capable of escaping from local optima” [97]. This technique uses perturbations or small probabilistic offsets to move the problem from a local maximization to a more global maximization. For this model, simulated annealing could be implemented using small offsets for short durations of coalition values to mitigate local maxima. However, I decided to examine the model results at this time without modifications. I will look to simulated annealing as a future modification of the model.

Game 12 presents a different challenge to the model. In game 12, 66.7% of the runs ended in successfully finding a core member. Analysis of the runs of this game indicates that the number of coalitions examined was likely insufficient to reach the appropriate result. To test this theory, I executed this game using a larger number of simulation ticks. 30 runs of the game were completed with a simulation time of 100,000 ticks. This new set of runs resulted in finding a core member 29 out of 30 times or 96.7% of the time.

My examination of the 13-agent game set identifies two potential concerns, local maximization and insufficient duration. Next, I examine the 15-agent games to determine if these findings are consistent and if they are likely to increase with an increasing agent-set size. Figure 21 shows the percentage of core coalition structure matches for the 15-agent hedonic games executed.

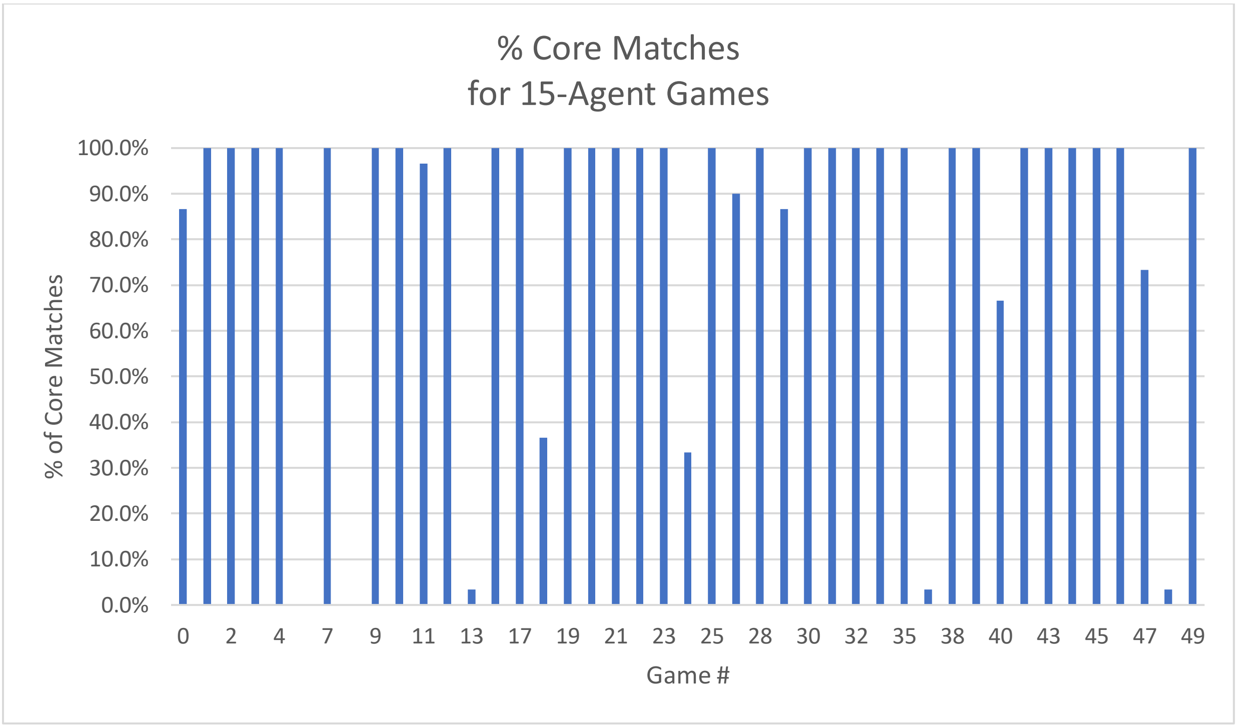


Figure 21: *Breakdown of 13-agent non-empty core game matches*

The 15-agent game set consists of 43 out of 50 games with a non-empty core. Of those 43 games, the ABCG model accurately selected a core coalition member 100% of the time for 30 unique games, failed to get a single correct coalition structure in 2 games, and achieved finding a core coalition only once in 3 games. A review of game 6, a game where a core coalition structure is never returned, shows that the ABCG model returned 24 unique coalition structures. This implies that the model was unable to reach stability in the game. This is most likely the result of insufficient variations of coalitions tested. Again, I attempt to increase the number of simulation ticks to 100,000. However, this did not improve the results. There were still 25 unique coalition structures and no core matches. It is still reasonable to assume that there were insufficient comparisons made. The number of possible coalition structures is 1,384,958,545; the average number of comparisons made by the algorithm for this game is 2,248,700. Doubling the number of simulation ticks increased the number of comparisons to 4,490,609 which still only amounts to about 0.325% of the total possible coalition structures.

The ABCG model performs extremely well for 12-agent sets or less and with moderate, mixed results for larger agent sets. Specifically, it performed well for the 14-agent set but not for the 13 and 15-agent sets. And this could be improved by increasing the number of simulation ticks executed for each game. And it accomplishes this with a relatively low number of comparisons required and much lower computation requirement than the naïve algorithm. Figure 22 shows the average number of comparisons made and the average duration of a game for each agent-set size. Despite the low number of comparisons in relation to the total number of possible coalition structures, the model is able to find a core member over 96% of the time when the core is not empty.

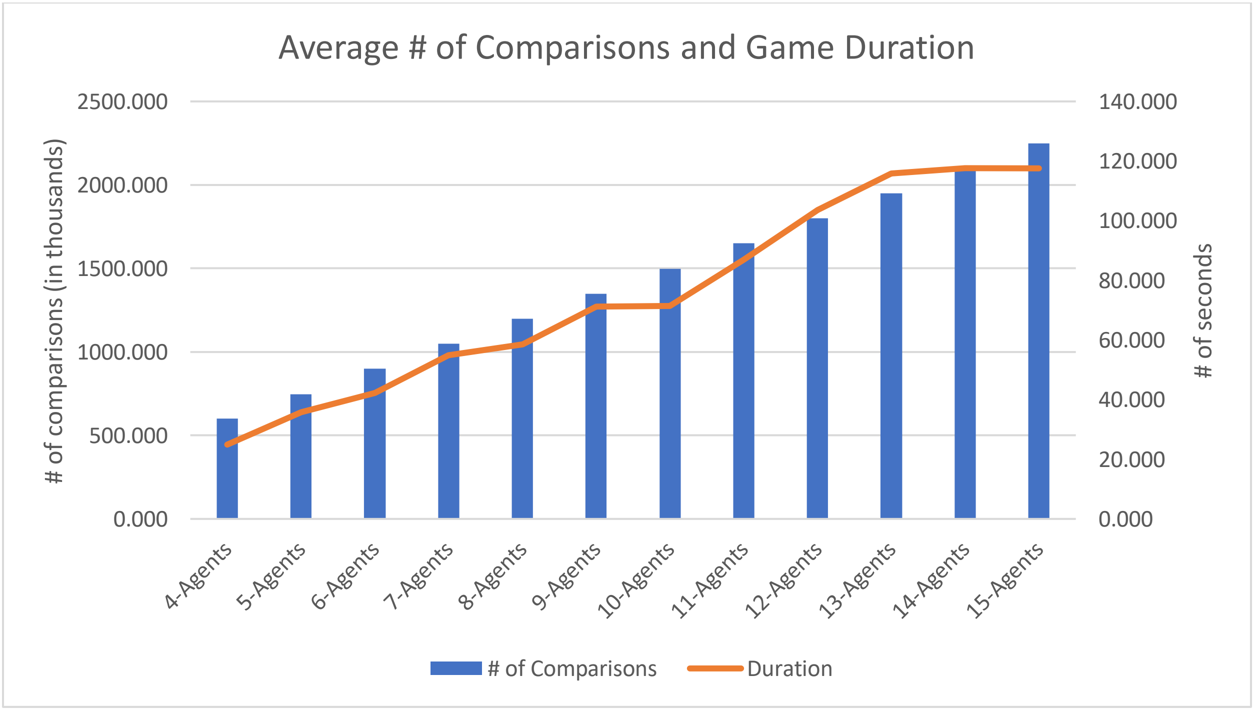


Figure 22:*Average # of comparisons performed and game duration for the ABCG model*

The trend for the subset shown depicts a roughly linear growth for both the time it took to execute the model and the number of comparisons made. While the problem set is bound at 15 agents due to the limitations in solving using the naïve algorithm, I believe that the trend would continue this path. However, there is evidence that the frequency with which the model can find a core member would decrease if the simulation time was held constant. An increase in the simulation time will not cause an exponential change in the model duration and might provide similar results for higher numbers of agents. This, however, is not empirically validated in this dissertation.

The computation requirements the ABCG model, unlike the naïve algorithm, does not grow uncontrollably with the increase in agent set size. The naïve algorithm is exhaustive and guaranteed to determine all members of the core solution. As previously mentioned, the computational worst case for the naïve algorithm is O (N^N). Figure 23 shows the time it took, in logarithmic scale, for the naïve algorithm to solve the existing games. While the naïve algorithm is far more efficient for small agent-set sizes, the ABCG model operates in significantly less time for the larger data sets. Additionally, the execution time and the number of comparisons can be varied, and the coalition structure returned is guaranteed to be at least independently rational.

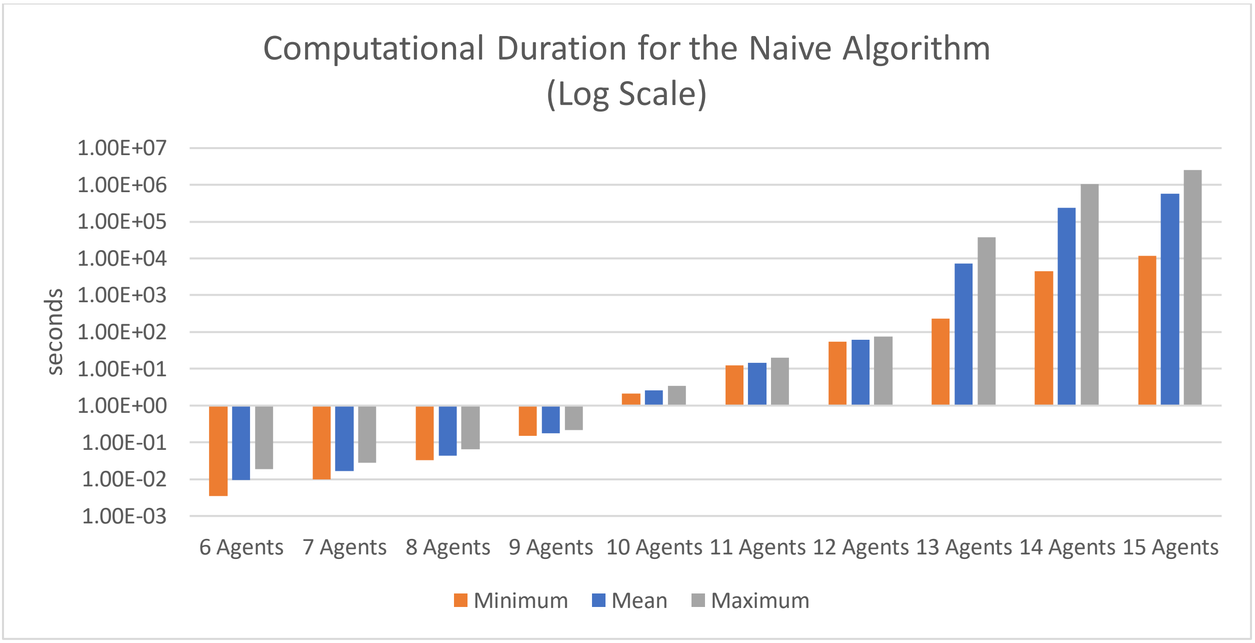


***Figure 23**: Time required to solve the hedonic games using the naive algorithm*

While I focus on games with a non-empty core, it is still important to understand that there are games with an empty core. In an experiment by Collins, Etemadidavan and Khallouli [98], they generated 1,000,000 hedonic games of agent sizes 13 or less. Table 15 shows their results for agent-set sizes 4 through 13. The average number of games with an empty core ranged from 1.96% for the 4-agent set to 7.21% for the 11-agent set, with an average of 5.73%.

***Table 15**: Percentage of games with an empty core in Collins et al. study*

% of Hedonic Games with an Empty Core										
4-Agent	5-Agent	6-Agent	7-Agent	8-Agent	9-Agent	10-Agent	11-Agent	12-Agent	13-Agent	Average
1.96%	3.36%	4.62%	5.70%	6.44%	6.84%	7.09%	7.21%	7.11%	6.97%	5.73%

The ABCG model produces a result irrespective of an empty or non-empty core. However, it means that the model cannot produce perfect results. The best the model can be expected to produce is subject to the expected level of non-empty core games. For example in Collins et al. [98], the best possible average for the hedonic games would be 94.27%. However, that does not imply that the model is useless if the core is empty. If the core is empty, the model will return a result that is at least individually rational. Unfortunately, unlike models that do not assume agents are selfishly rational, the ABCG model cannot claim that each iteration improves the social welfare (the sum of all coalition values of the coalition structure). Social welfare maximization is not a function of core solutions.

Experiment #2 shows the potential of the model. The positive outcomes are that for games with a non-empty core, the model is very effective – it correctly identifies a core coalition structure 99.5% of the time – for games with 12 agents or less and showed some level of effectiveness for agent-set sizes 13, 14 and 15; 88.3%, 93.6%, and 79.6% respectively. These results are most likely due to two factors: local maximization and insufficient number of comparisons. Local maximization can be mitigated, but not eliminated, using simulated annealing. Insufficient comparisons can be overcome by increasing the number of simulation ticks for which the model is executed. This is a reasonable substitute for the naïve algorithm as the duration and number of comparisons for the results gained are considerably less than the full spectrum required for an exhaustive search.

The model also always returns a coalition structure. At a minimum, the coalition structure is always individually rational. Further, even with a comparatively small search compared to the possible coalition structures the model was able to return a coalition structure that was a core member more than half the time (including evaluating games with an empty core solution set). Future work with this model will include determining adequate simulation time based on the size of the game. This will provide a useful boundary for varying agent-set sizes.

Experiment #2 showed several positive outcomes of the ABCG model. Experiment #3 is designed to determine if the initial coalition structure has any impact on the final coalition structure. Sandholm et al. [30] note that for determining the social welfare maximization – i.e.

maximizing the sum of all coalitions – it is imperative to examine the grand coalition – all agents in a single coalition – and his level 2 coalition structure (described in the literature review). Experiment #3 is structured to explore any differences that occur when the initial coalition structure is not pre-set to the grand coalition and the grand coalition is not explicitly tested. The model is expected to perform the same regardless of which the initial coalition to which the agents belong.

4.1.3 EXPERIMENT #3

Experiment #3 provides an opportunity to determine if the initial coalition structure provides any advantages or disadvantages to the model results. That is, I examine whether the model is sensitive to the initial condition of the coalition structure. Experiment #2 has each model run begin with the agents all in singleton groups. The total number of coalitions in the coalition structure was equal to the number of agents in the game. In Experiment #3 I first initialized each run with all agents in one coalition – the grand coalition. I determine the percentage of times a core member is correctly identified and compare the results of the initial condition being the singleton coalitions to the initial condition being the grand coalition. The comparison allows me to review any differences that occur due to the initial condition. I then randomly assign coalitions for the initial coalition structure and determine the percentage of times a core member is found. Agents are assigned a coalition number from zero to the maximum number of agents. All agents with the same coalition number are placed in the same coalition without regard to preference to initial the game. The purpose of this comparison is to determine whether the model is sensitive to any initial condition. If the results of all three initial coalition setups yield similar results when finding a core member, then I conclude that the model is not sensitive to the initial condition of the coalition structure.

Figure 24 shows the comparison for each of the three initial comparisons for all the games. The games used for this experiment all have a non-empty core. There is minimal difference between each of the samples based on the initialization of the coalition structure. However, to ensure that there are no statistically significant differences among the samples, I performed paired tests of the samples for comparison.

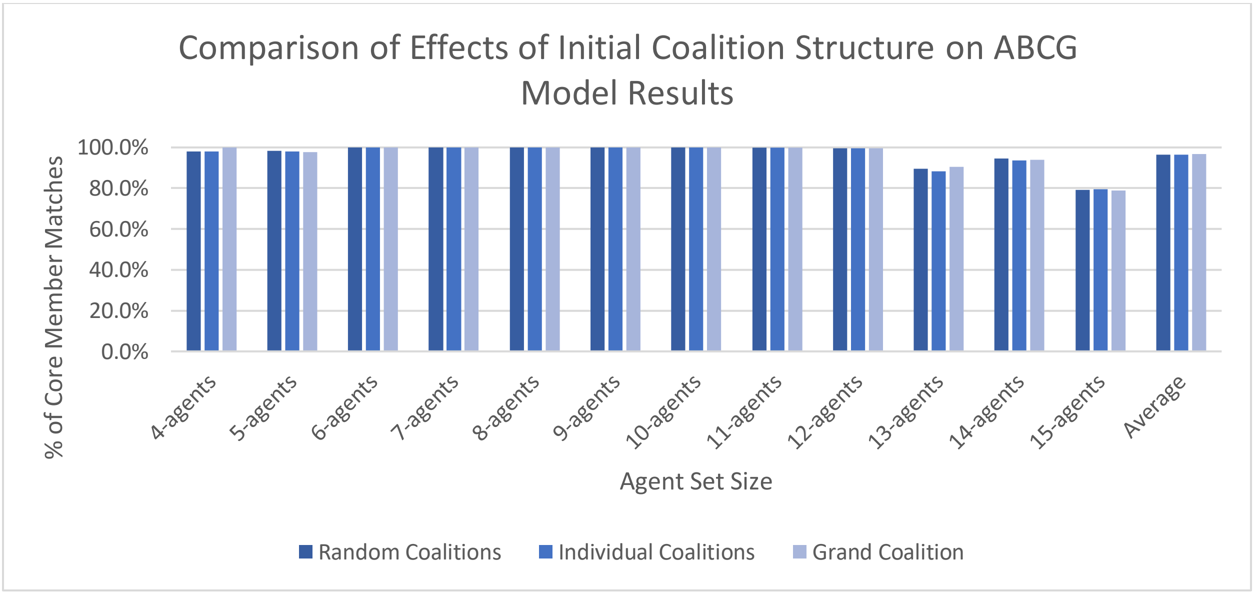


Figure 24: *Percentage of core member matches for various initial coalition structures*

Table 16 shows the percentage of core matches for each of the experiment samples and the differences between each of the paired samples.

Table 16:*Percentage of core matches for different initial coalition structures*

# of Agents	Random Coalitions	Individual Coalitions	Grand Coalition	Random to Individual	Random to Grand
4-agents	97.96%	98.03%	100.00%	-0.001	-0.020
5-agents	98.22%	98.00%	97.85%	0.002	0.004
6-agents	100.00%	100.00%	100.00%	0.000	0.000
7-agents	100.00%	100.00%	100.00%	0.000	0.000
8-agents	100.00%	100.00%	100.00%	0.000	0.000
9-agents	100.00%	100.00%	100.00%	0.000	0.000
10-agents	100.00%	100.00%	100.00%	0.000	0.000
11-agents	99.79%	99.79%	99.93%	0.000	-0.001
12-agents	99.62%	99.70%	99.55%	-0.001	0.001
13-agents	89.60%	88.33%	90.40%	0.013	-0.008
14-agents	94.47%	93.58%	93.90%	0.009	0.006
15-agents	79.19%	79.56%	78.81%	-0.004	0.004
Average	96.57%	96.41%	96.70%	0.002	-0.001

I performed a series of paired two-tailed t tests to determine if there is a statistically significant difference between starting the model with each agent in a singleton coalition versus starting the model with agents in randomly assigned coalitions. I then performed the same test with the initial coalition structure of all the agents in a single coalition – the grand coalition. The paired t test is used to determine if the mean difference between the sets of results is zero. Like many statistical tests, the two-tailed test has underlying assumptions. The first two assumptions are that the data is independent and identically distributed (IID); the third assumption is that the data distribution is normal. At this point I note the deficiency of performing this test. The data is not IID nor is it normally distributed. The data for each of the data initialization groups is negatively skewed indicating a consistent departure from a normal distribution. The data is also bounded – it cannot exceed 100% and cannot go below 0%. However, the nature of the paired test allows us to glean useful information despite relaxing the underlying conditions. Table 3 provides some descriptive statistics that provide an understanding of similarities and differences in the data.

The null hypothesis is that there is no difference between the paired samples: $H_0: \mu_1 - \mu_2 = 0$ and the alternative hypothesis is that there is a difference between the paired samples: $H_a: \mu_1 - \mu_2 \neq 0$. The significance level I use for rejecting the null hypothesis is 0.01. When I compare the results of the randomly initialized coalition structure to the individual coalitions in the initial coalition structure, I determine that the probability of the mean difference being zero is less than the significance level. Therefore, I accept the null hypothesis. The same is true for the randomly initialized to the initialization with the grand coalition structure. Again, I accept the null hypothesis and conclude that the initial coalition structure does not have a statistically significant impact on the results of the model.

Table 17:*T-Test results for the mean comparison of the null and alternative hypotheses*

	Random to Individual Coalition Initialization	*Random to Grand Coalition Initialization*
Pearson Correlation	0.9975	0.9943
Hypothesized Mean Difference	0	0
df	11	11
t Stat	1.1776	-0.6674
P(T<=t) two-tail	0.2638	0.5183
t Critical two-tail	3.1058	3.1058

The previous experiments showed me how the initial model compared to the model found in the literature, how the modified model performed against empirical data, and how the initial condition impacted the model. Experiment #4 shows how the model can be used further in research.

4.1.4 EXPERIMENT #4

Experiment #4 demonstrates the workings of the model in a common research use case. Similar to experiment #1, the use case is a variation on the Glove Game created by Shapley and

Shubik [72] to represent a simple assignment market. In this variation, I randomly assign both left gloves and right gloves to each agent. I also randomly assign a preference for either total numbers of gloves or pairs of gloves. I utilized hedonic games as they have the potential to represent various types of games. In this case, I tally the gloves, left and right respectively, and divide them among the agents. Each agent's coalition value or utility is based on their preference for total gloves vs pairs of gloves. Specifically, if an agent prefers pairs, the agent's value for the coalition is the total number of pairs divided by the number of agents in the coalition. If the agent's preference is for total gloves, half the total number of gloves in the coalition is divided by the number of coalition members to obtain the agent's coalition value.

The primary differences between the strict hedonic game and the glove game are the method in which the coalition value is derived. All other factors are the same. This allows us to utilize the ABCG model for both types of games. The data is generated using a naïve algorithm like the strict hedonic game that tests every coalition structure to find a member of the core. Agents are initialized with a random preference for pairs vs total gloves and a random number, between zero and nine, of right gloves and left gloves independently. The coalition value array is created, and the program runs the same as previously.

As mentioned in the data section, my glove game produced a non-empty core about 23.5% of the time. The data used is only for the non-empty core games. Due to the extreme duration of these runs and the low percentage of games with a non-empty core, I decide to reduce the number of unique games to 20. With 30 stochastic runs on 20 games, this creates a sample of 600 games. I believe that this level of sampling does not degrade the statistical power.

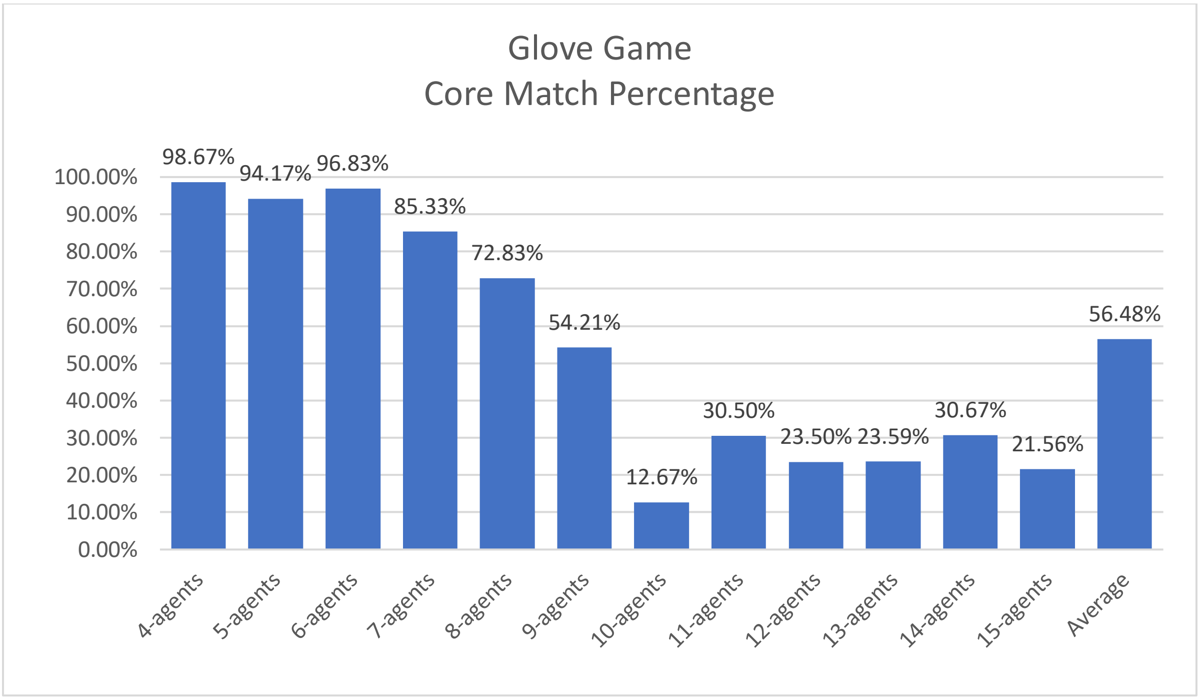


Figure 25: *Percentage of core matches for glove game*

Figure 25 shows the frequency with which the model identifies a core member in this assignment game type structure for the given games. The frequency is lower than in strict hedonic game experiments. As I examine both the glove games, I note that there is wide variation in the size of the core both within and between agent-set sizes. The core sizes range from 1 to 3,520. This greater variation has produced more local maximization and a need for examining more coalition combinations. There is a steady decline in the ability to find a core coalition structure as the agent-set size increases. As I have seen in the previous experiments, I recognize that there is possibly a need for more iterations through executing more simulation ticks.

Although the frequency is lower than expected, I believe that this primarily the effect of local maximization. To describe the impact of this effect, I examine a game in the 8-agent game set. Out of the 30 stochastic runs, the ABCG model correctly identified a core member 8 times or 26.7% of the time. Inspection of this game shows that 28 coalitions out of a possible 255 coalitions meet the individual rationality clause. However, of those combinations the values for each agent in the different combinations are often the same. For example, Agent 7 is a member of 11 of the 28 individually rational coalitions. Of those 11 coalitions, the value for Agent 7 is the same for nine of them. The model reaches a local maximization often for coalitions involving Agent 7 because this agent has no incentive to change coalitions. This is a limitation of my model. It is considerably less effective when there is significant similarity in the coalition values of an agent. In this example, the use of glove resources between zero and nine created highly similar coalition values in the data. Expressing the utility value in a broader spectrum will help alleviate, but not eliminate, this local maximization problem.

Another challenge to my model is in the experimentation design. I arbitrarily determined the number of simulation ticks for which the model was executed. The uncertainty of reaching steady state caused us to question the results. To eliminate the uncertainty of this, the experiment would have benefitted from an improved method of determining the necessary duration. While smaller agent-set sizes can achieve steady-state in relatively few timesteps, increases in the agent set size require that I revisit the number of iterations needed to reach a steady state. The relation between number of time steps and the size of the agent-set is worthy of its own experiment and should be considered in future work.

The use of the ABCG model for the traditional assignment game requires further refinement but still provides significant benefits. A core coalition structure was found 56.48% of the time. For the approximately 77% of the empty core games, the model produces a coalition structure that is at least individually rational. Any coalitions within the coalition structure that is not a singleton coalition is an improvement over the individually rational singleton coalition for each agent in the coalition. This provides some benefits to the researcher for all agent-set sizes.

4.2 SUMMARY

In this chapter, I reviewed the series of hedonic games that comprise the data set for my experimentation. I performed four sets of experiments designed to compare my model to an existing model, statistically validated the frequency with which my model was able to find a core member, determined the impact of the initial condition on the model results, and showed the usefulness of the model in a traditional research problem.

I demonstrated how the naïve algorithm created strict hedonic games and solved for those games for the core solution. I identified games with an empty core vs a non-empty core. And I described the amount of time that solving the core required including the exponential growth in duration time for the solutions as the agent-set size increased.

I compared the ABCG model to a model in the existing literature and demonstrated that it outperformed the current model. I detailed the results of the experiments and showed that the ABCG model can find a core member greater than 96% of the time for games with a non-empty core solution. The challenges I noted with the model was local maximization and the need to better identify the number of iterations required. The local maximization is often a challenge when implementing heuristics. The downside to heuristic modeling is that without a central authority, local maximizations can occur. I performed a sensitivity analysis and determined that the initial condition that the agents were in had no impact on the model’s performance. Finally, I showed how the model can be used to with a traditional research problem. I took a modified version of the glove game and used the ABCG model to find a core member. The model was only successful in finding a core member 56.48% of the time. While the model was not as successful on the glove game as anticipated, it highlighted the need to understand the variance in the data and determine the linkage between the agent-set size and the number of time step iterations needed.

In the final chapter, I provide my conclusion of this work and discuss the work I have planned for the future.

5.0 CONCLUSION AND FUTURE WORK

One of the topics that have always fascinated me is understanding why groups form the way they do. In classroom settings, we are often required to break into groups. In business, politics, and military strategy, we form alliances. There are a finite number of potential partners in these groupings but an exceptional number of possibilities. The study of strategic coalition formation is conceptually simple. There are a finite number of possible coalitions and agents have their preferences for which coalition they would like to join. The preference for one coalition over another, whether the result of resource access or joy of community, can be expressed as a utility value the agent holds for that coalition. And agents form coalitions based on those values.

As mentioned, conceptually this is simple. However, it is tremendously difficult to actualize for any significant number of agents. Each agent's preference is unique and combining these unique preferences to find a stable structure requires an enormous amount of computational effort. The time study in Experiment #2 shows that while checking each coalition structure for an agent-set size of 10 can be performed in an average 2.6 seconds, increasing the number of agents to 15 raises the average computation time to 579,757.61 seconds (or 161 hours). This makes the analytical solution difficult to use. Faster processors and parallel efforts can reduce the time, but this also comes at a cost. The motivation for this research is to introduce a heuristic method that can aid researchers in the ability to study of coalition formation through current economic theory without the computational overhead.

Previous work on this topic shows that coalition formation is most often studied using cooperative game theory. Agent-based modeling and multi-agent systems have attempted to incorporate some aspects of cooperative game theory. This indicates that there is an understanding of the ability to ABM to be an effective tool in studying coalition formation. However, the models do not provide complete representation of the theory, nor do they demonstrate complete alignment of the theory to the practical model. In this work, I have provided a heuristic that closely aligns with cooperative game theory and empirically tested the model to determine if it can obtain a coalition structure that is a member of the core greater than 90% of the time for 15 agents or less. The model achieved a degree of success. Using strict hedonic games, the model found a core coalition structure over 96% of the time and the initial coalition structure did not statistically impact this result. Using the modified glove game was more challenging for the model. The glove game allowed ties in the coalition values. This created challenges in producing games with non-empty cores (only about 23% of the games had a non-empty core) and with the model's ability to find a stable coalition structures. The model was only able to reach a core coalition structure about 58% of the time.

This dissertation intends to advance modeling and simulation by providing a heuristic that aids in studying strategic coalition formation using CGT. It delivers a model that improves on the model found in the literature, provides an efficient computational structure for comparing coalition values, and provides a tool that can be expanded for solution concepts beyond the core solution. For example, to use the model for social welfare, the only adjustment to the model would be the comparison values. Likewise to incorporate Shapley's [59] marginal contributions, the adjustment required would be to re-initialize the coalition value matrix to reflect each agent's payoff. The interaction structure and mechanisms to retrieve the coalition values remains unchanged.

While the model produced mixed results, it demonstrates opportunities to future development as well as future research. The two areas of focus for improving the model are understanding the proper duration or simulation ticks with respect to the agent-set size and the impact of similar utility values for the various coalitions. Both these topics can be explored with additional experimentation. However, further exploration of the topic can hopefully generate a more dynamic model that incorporates changes in coalition values based on dynamic events. For example, if a modeler needed to explore how the coalition dynamics are changing in the current environment with respect to the conflict in the Ukraine, the model would provide a platform to represent the changes in coalitions and coalition favor based on the population sentiment.

The field of strategic coalition formation is fertile for exploration with modeling and simulation. Heuristic algorithms provide an aid to traditional computational methods and the various social science fields have already embraced the discipline in many areas, for example Computational Social Science and Agent-based Computational Economics. There are many opportunities to formalize more models of their theories.

REFERENCES

1. Smirnov, A. and L. Sheremetov, *Models of coalition formation among cooperative agents: The current state and prospects of research.* Scientific and Technical Information Processing, 2012. **39**(5): p. 283-292.
2. Axtell, R., *Why agents?: on the varied motivations for agent computing in the social sciences.* 2000.
3. Epstein, J.M. and R. Axtell, *Artificial societies and generative social science.* Artificial Life and Robotics, 1997. **1**(1): p. 33-34.
4. Gilbert, N. and S. Bankes, *Platforms and methods for agent-based modeling.* Proceedings of the National Academy of Sciences, 2002. **99**(suppl 3): p. 7197-7198.
5. Chalkiadakis, G., E. Elkind, and M. Wooldridge, *Computational aspects of cooperative game theory.* Synthesis Lectures on Artificial Intelligence and Machine Learning, 2011. **5**(6): p. 1-168.
6. Gillies, D.B., *Solutions to general non-zero-sum games.* Contributions to the Theory of Games, 1959. **4**(40): p. 47-85.
7. Ballester, C., *NP-completeness in hedonic games.* Games and Economic Behavior, 2004. **49**(1): p. 1-30.
8. Bell, E.T., *The iterated exponential integers.* Annals of Mathematics, 1938: p. 539-557.
9. Von Neumann, J. and O. Morgenstern, *Theory of Games and Economic Behavior, 2nd rev*. 1947: Princeton.
10. Aumann, R.J. and J.H. Dreze, *Cooperative games with coalition structures.* International Journal of Game Theory, 1974. **3**(4): p. 217-237.
11. Peleg, B. and P. Sudhölter, *Introduction to the theory of cooperative games*. Game Theory, Mathematical Programming and Operations Research. 2003, Boston, MA: Kluwer Academic Publishers.
12. Rahwan, T., et al. *Anytime optimal coalition structure generation*. in *AAAI*. 2007.
13. Sung, S.C. and D. Dimitrov, *On core membership testing for hedonic coalition formation games.* Operations Research Letters, 2007. **35**(2): p. 155-158.
14. Shehory, O. and S. Kraus, *Methods for task allocation via agent coalition formation.* Artificial Intelligence, 1998. **101**(1-2): p. 165-200.
15. Collins, A.J. and E. Frydenlund. *Agent-based modeling and strategic group formation: A refugee case study*. in *Winter Simulation Conference (WSC), 2016*. 2016. IEEE.
16. Bonnevay, S., N. Kabachi, and M. Lamure. *Agent-based simulation of coalition formation in cooperative games*. in *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*. 2005. IEEE.
17. Janovsky, P. and S.A. DeLoach. *Multi-agent simulation framework for large-scale coalition formation*. in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 2016. IEEE.
18. Farinelli, A., et al., *A hierarchical clustering approach to large-scale near-optimal coalition formation with quality guarantees.* Engineering Applications of Artificial Intelligence, 2017. **59**: p. 170-185.
19. Cronbach, L.J., *Coefficient alpha and the internal structure of tests.* psychometrika, 1951. **16**(3): p. 297-334.
20. Gamson, W.A., *A theory of coalition formation.* American Sociological Review, 1961: p. 373-382.
21. Laver, M. and K.A. Shepsle, *Coalitions and Cabinet Government.* American Political Science Review, 1990. **84**(3): p. 873-890.

22. Bäck, H., *Intra-Party Politics and Coalition Formation: Evidence from Swedish Local Government.* Party Politics, 2008. **14**(1): p. 71-89.
23. Holler, M.J., *Forming Coalitions and Measuring Voting Power.* Political Studies, 1982. **30**(2): p. 262-271.
24. Jackson, M.O. and B. Moselle, *Coalition and party formation in a legislative voting game.* Journal of Economic Theory, 2002. **103**(1): p. 49-87.
25. Silk, J.B., S.C. Alberts, and J. Altmann, *Patterns of coalition formation by adult female baboons in Amboseli, Kenya.* Animal Behaviour, 2004. **67**(3): p. 573-582.
26. Thierry, B., *Primate socioecology, the lost dream of ecological determinism.* Evolutionary Anthropology: Issues, News, and Reviews, 2008. **17**(2): p. 93-96.
27. Janson, C.H., *Evolutionary ecology of primate social structure*. Evolutionary ecology and human behavior. 2017: Routledge. 95-130.
28. Rahwan, T. and N.R. Jennings, *An improved dynamic programming algorithm for coalition structure generation*, in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. 2008, International Foundation for Autonomous Agents and Multiagent Systems. p. 1417-1420.
29. Rahwan, T., et al., *An anytime algorithm for optimal coalition structure generation.* Journal of Artificial Intelligence Research, 2009. **34**: p. 521-567.
30. Sandholm, T., et al., *Coalition structure generation with worst case guarantees.* Artificial Intelligence, 1999. **111**(1-2): p. 209-238.
31. Levy, J.S., *Alliance formation and war behavior: An analysis of the great powers, 1495-1975.* Journal of Conflict Resolution, 1981: p. 581-613.
32. Kahan, J.P. and A. Rapoport, *Theories of coalition formation*. 2014: Psychology Press.
33. Ghemawat, P., M.E. Porter, and R.A. Rawlinson, *Patterns of international coalition activity.* Competition in global industries, 1986: p. 345-366.
34. Posen, B.R., *ESDP and the Structure of World Power.* The international spectator, 2004. **39**(1): p. 5-17.
35. Gilbert, N., *Agent-based models*. 2008: Sage.
36. Macal, C.M. and M.J. North. *Agent-based modeling and simulation: Desktop ABMS*. in *2007 Winter Simulation Conference*. 2007. IEEE.
37. Collins, A.J. and E. Frydenlund, *Strategic group formation in agent-based simulation.* Simulation, 2018. **94**(3): p. 179-193.
38. Dreze, J.H. and J. Greenberg, *Hedonic coalitions: Optimality and stability.* Econometrica: Journal of the Econometric Society, 1980: p. 987-1003.
39. Aumann, R.J. and B. Peleg, *Von Neumann-Morgenstern solutions to cooperative games without side payments.* Bulletin of the American Mathematical Society, 1960. **66**(3): p. 173-179.
40. Macal, C.M. and M.J. North. *Tutorial on agent-based modeling and simulation*. in *Proceedings of the Winter Simulation Conference, 2005.* 2005. IEEE.
41. Bonabeau, E., *Agent-based modeling: Methods and techniques for simulating human systems.* Proceedings of the national academy of sciences, 2002. **99**(suppl 3): p. 7280-7287.
42. Railsback, S.F. and V. Grimm, *Agent-based and individual-based modeling: a practical introduction*. 2019: Princeton university press.
43. Ferber, J., O. Gutknecht, and F. Michel. *From agents to organizations: an organizational view of multi-agent systems*. in *International workshop on agent-oriented software engineering*. 2003. Springer.
44. Epstein, J.M. and R. Axtell, *Growing artificial societies: social science from the bottom up*. 1996: Brookings Institution Press.

45. Schelling, T.C., *Dynamic models of segregation†.* Journal of Mathematical Sociology, 1971. **1**(2): p. 143-186.
46. Thomas, L.C., *Games, theory and applications*. 2012: Courier Corporation.
47. Schulz, A.S. and N.A. Uhan, *Approximating the least core value and least core of cooperative games with supermodular costs.* Discrete Optimization, 2013. **10**(2): p. 163-180.
48. Harsanyi, J.C. and R. Selten, *A general theory of equilibrium selection in games.* MIT Press Books, 1988. **1**.
49. Karakaya, M., *Hedonic coalition formation games: A new stability notion.* Mathematical Social Sciences, 2011. **61**(3): p. 157-165.
50. Bogomolnaia, A. and M.O. Jackson, *The stability of hedonic coalition structures.* Games and Economic Behavior, 2002. **38**(2): p. 201-230.
51. Iehlé, V., *The core-partition of a hedonic game.* Mathematical Social Sciences, 2007. **54**(2): p. 176-185.
52. Tohmé, F. and T. Sandholm, *Coalition formation processes with belief revision among bounded-rational self-interested agents.* Journal of Logic and Computation, 1999. **9**(6): p. 793-815.
53. Rahwan, T., *Algorithms for coalition formation in multi-agent systems*. 2007, University of Southampton.
54. De Bruijn, N.G., *Asymptotic methods in analysis*. Vol. 4. 1981: Courier Corporation.
55. Yeh, D.Y., *A dynamic programming approach to the complete set partitioning problem.* BIT Numerical Mathematics, 1986. **26**(4): p. 467-474.
56. Rothkopf, M.H., A. Pekeč, and R.M. Harstad, *Computationally manageable combinational auctions.* Management Science, 1998. **44**(8): p. 1131-1147.
57. Ramchurn, S.D., et al. *Coalition formation with spatial and temporal constraints*. in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 3-Volume 3*. 2010. International Foundation for Autonomous Agents and Multiagent Systems.
58. Zolezzi, J.M. and H. Rudnick, *Transmission cost allocation by cooperative games and coalition formation.* IEEE Transactions on power systems, 2002. **17**(4): p. 1008-1015.
59. Shapley, L.S., *Stochastic games.* Proceedings of the National Academy of Sciences, 1953. **39**(10): p. 1095-1100.
60. Banzhaf III, J.F., *Weighted voting doesn't work: A mathematical analysis.* Rutgers Law Review, 1964. **19**: p. 317.
61. Browne, E.C. and J.P. Frendreis, *Allocating coalition payoffs by conventional norm: an assessment of the evidence from cabinet coalition situations.* American Journal of Political Science, 1980: p. 753-768.
62. Crott, H.W. and W. Albers, *The equal division kernel: An equity approach to coalition formation and payoff distribution in N-person games.* European Journal of Social Psychology, 1981. **11**(3): p. 285-305.
63. Nash, J., *Non-cooperative games.* Annals of Mathematics, 1951: p. 286-295.
64. Mustafee, N., et al. *Purpose and benefits of hybrid simulation: contributing to the convergence of its definition*. in *2017 Winter Simulation Conference (WSC)*. 2017. IEEE.
65. Mustafee, N. and J.H. Powell. *From hybrid simulation to hybrid systems modelling*. in *2018 Winter Simulation Conference (WSC)*. 2018. IEEE.
66. Howick, S. and F. Ackermann, *Mixing OR methods in practice: Past, present and future directions.* European Journal of Operational Research, 2011. **215**(3): p. 503-511.
67. Szilagyi, M.N., *The El Farol Bar Problem as an Iterated N-Person Game.* Complex Systems, 2012. **21**(2): p. 153.
68. Arthur, W.B., *Inductive reasoning and bounded rationality.* The American Economic Review, 1994. **84**(2): p. 406-411.

69. Collins, A.J. *Strategically Forming Groups in the El Farol Bar Problem*. in *Proceedings of the 2017 International Conference of The Computational Social Science Society of the Americas*. 2017. ACM.
70. Watson, R., *Quantitative research.* Nursing Standard, 2015. **29**(31).
71. Windrum, P., G. Fagiolo, and A. Moneta, *Empirical validation of agent-based models: Alternatives and prospects.* Journal of Artificial Societies and Social Simulation, 2007. **10**(2): p. 8.
72. Shapley, L.S. and M. Shubik, *The assignment game I: The core.* International Journal of game theory, 1971. **1**(1): p. 111-130.
73. Matsumoto, M. and T. Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.* ACM Transactions on Modeling and Computer Simulation (TOMACS), 1998. **8**(1): p. 3-30.
74. Chalkiadakis, G., et al. *Cooperatives of distributed energy resources for efficient virtual power plants*. in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. 2011. International Foundation for Autonomous Agents and Multiagent Systems.
75. Collins, A.J., S. Etemadidavan, and W. Khallouli, *Generating Empirical Core Size Distributions of Hedonic Games using a Monte Carlo Method.* arXiv preprint arXiv:2007.12127, 2020.
76. Djokić, B., et al., *Short Note: A Fast Iterative Algorithm for Generating Set Partitions.* The Computer Journal, 1989. **32**(3): p. 281-282.
77. Metropolis, N. and S. Ulam, *The monte carlo method.* Journal of the American statistical association, 1949. **44**(247): p. 335-341.
78. Sobol, I.M., *A primer for the Monte Carlo method*. 1994: CRC press.
79. Wilensky, U., *NetLogo. Evanston, IL: center for connected learning and computer-based modeling, Northwestern University*. 1999.
80. Grimm, V., et al., *The ODD protocol: a review and first update.* Ecological modelling, 2010. **221**(23): p. 2760-2768.
81. Collins, A., et al., *A call to arms: standards for agent-based modeling and simulation.* Journal of Artificial Societies and Social Simulation, 2015. **18**(3): p. 12.
82. Hart, S. and M. Kurz, *Endogenous formation of coalitions.* Econometrica: Journal of the econometric society, 1983. **51**(4): p. 1047-1064.
83. Banerjee, S., H. Konishi, and T. Sönmez, *Core in a simple coalition formation game.* Social Choice and Welfare, 2001. **18**(1): p. 135-153.
84. Thomas, L.C., *Games, Theory and Applications*. 2003, Mineola, NY: Dover Publications.
85. Sokolowski, J.A. and C.M. Banks, *Modeling and simulation fundamentals: theoretical underpinnings and practical domains*. 2010: John Wiley & Sons.
86. Law, A.M., W.D. Kelton, and W.D. Kelton, *Simulation modeling and analysis*. Vol. 2. 1991: McGraw-Hill New York.
87. Balci, O., *Validation, verification, and testing techniques throughout the life cycle of a simulation study.* Annals of operations research, 1994. **53**(1): p. 121-173.
88. Klügl, F. *A validation methodology for agent-based simulations*. in *Proceedings of the 2008 ACM symposium on Applied computing*. 2008. ACM.
89. Axtell, R., et al., *Aligning simulation models: A case study and results.* Computational & mathematical organization theory, 1996. **1**(2): p. 123-141.
90. Mendenhall, W.M. and T.L. Sincich, *Statistics for Engineering and the Sciences*. 2016: CRC Press.
91. Cohen, J., *Statistical power analysis.* Current directions in psychological science, 1992. **1**(3): p. 98-101.
92. Erdfelder, E., F. Faul, and A. Buchner, *GPOWER: A general power analysis program.* Behavior research methods, instruments, & computers, 1996. **28**(1): p. 1-11.

93. Hart, S. and M. Kurz, *Endogenous formation of coalitions.* Econometrica: Journal of the Econometric Society, 1983: p. 1047-1064.
94. Senge, P.M. and J.W. Forrester, *Tests for building confidence in system dynamics models.* System dynamics, TIMS studies in management sciences, 1980. **14**: p. 209-228.
95. van Laarhoven, P. and E. Aarts, *Simulated Annealing. Eindhoven, Netherlands: D.* Reidel Publishing Company, 1987. **10**: p. 978-94.
96. Kirkpatrick, S., C.D. Gelatt Jr, and M.P. Vecchi, *Optimization by simulated annealing.* science, 1983. **220**(4598): p. 671-680.
97. Henderson, D., S.H. Jacobson, and A.W. Johnson, *The theory and practice of simulated annealing*, in *Handbook of metaheuristics*. 2003, Springer. p. 287-319.
98. Collins, A.J., S. Etemadidavan, and W. Khallouli, *Generating empirical core size distributions of hedonic games using a Monte Carlo Method.* International Game Theory Review, 2021: p. 2250001.

APPENDICES

APPENDIX: NAÏVE ALGORITHM FOR DETERMINING CGT CORE

```
/* Brute Force Coalition Formation    */

/* Daniele Vernon-Bido              */

/* November 10, 2019   Initial version*/


/* Glove Game Version August 22, 2020 */


/* Main Program */


#include<iostream>

#include<fstream>

#include<sstream>

#include<iterator>

#include<string>

#include<cmath>

#include<algorithm>

#include<ctime>

#include<chrono>


using namespace std;

using namespace std::chrono;


/* Global Variables */

float **colArray = NULL;            /* Contains the hedonic preference values*/

int *singletonArray = NULL;        /* Contains the preference for remaining alone */

int *colStructureValue = NULL;               /* Contains the preference for each agent in a coalition
structure */
```

```
int *blockedCoalitionArray = NULL;

int numAgents;

int numCoalitions;

string inFile;

string outFile;

string fileAppend;



int **resourceArray = NULL;        /* Contains the individual preference, # of left gloves, # of right gloves
*/


string ConvertToBinary(unsigned long n)

{

            char    result[(sizeof(unsigned long) * 8) + 1];

            unsigned index = sizeof(unsigned long) * 8;

            result[index] = '\0';


            do result[--index] = '0' + (n & 1);

            while (n >>= 1);


            string bin = string(result + index);


            /* Set string to proper length */

            if (bin.length() < numAgents)

                        bin.insert(0, numAgents - bin.length(), '0');


            return bin;

}


void SetRandomCoalitionValues() {
```

```
/* DETERMINE COALITION VALUES*/


string pFile;
string pString;
pFile.append("Input");
pFile.append(fileAppend);
cout << pFile << endl;


ofstream preferenceFile(pFile);
preferenceFile.open(pFile, ios::out);
if (preferenceFile.is_open())
        preferenceFile.clear();


/* Test write file */
if (preferenceFile.fail())
        cout << "fail bit set" << endl;
if (preferenceFile.eof())
        cout << "eof bit set" << endl;
/**/



int *hashArray = NULL;
int c = pow(2, numAgents);      /* c is the number of possible coalitions */
unsigned int x = numAgents;
int m = pow(2, numAgents - 1);   /* m is max number of coalitions in which an agent can exist */
string bString;                 /* bString holds the converted bit string */
int leftGloves;
int rightGloves;
int coalitionAgents;
```

```
	/* Dynamic 2D resource array allocation */
	resourceArray = new int *[numAgents];
	for (int i = 0; i < numAgents; i++) {
		resourceArray[i] = new int[3];
		//for (int j = 0; j < 3; j++) {
			resourceArray[i][0] = rand() % 2;
			resourceArray[i][1] = rand() % 10;	/* pick a number between 0 and 9 to
allocate # of left gloves */
			resourceArray[i][2] = rand() % 10;	/* pick a number between 0 and 9 to
allocate # of right gloves */
		//}
	}


	/* Dynamic 2D array allocation & coalition value assignment */
	colArray = new float *[numCoalitions];
	for (int i = 0; i < numCoalitions; i++) {
		leftGloves = 0;
		rightGloves = 0;
		coalitionAgents = 0;
		colArray[i] = new float[numAgents];
		for (int j = 0; j < numAgents; j++) {
			colArray[i][j] = 0;
			bString = ConvertToBinary(i);
			if (bString.length() < numAgents)
				bString.insert(0, numAgents - bString.length(), '0');
			if (bString[j] == '1') {
				leftGloves = leftGloves + resourceArray[j][1];
				rightGloves = rightGloves + resourceArray[j][2];
				coalitionAgents++;
			}
		}
		if (coalitionAgents > 0) {
			float prefOne = (min(leftGloves, rightGloves)) / coalitionAgents;
			float prefTwo = ((leftGloves + rightGloves) * .5) / coalitionAgents;
			for (int j = 0; j < numAgents; j++) {
				if (bString[j] == '1') {
					if (resourceArray[j][0] == 0) {
						colArray[i][j] = prefOne;
					}
					else {
						colArray[i][j] = prefTwo;
					}
				}
			}
		}

	}



	cout << "Write to Preference File" << endl;


	for (int i = 0; i < numAgents; i++) {
		pString.clear();
		for (int j = 0; j < 3; j++) {
			pString.append(to_string(resourceArray[i][j]));
			if (j < 2)
				pString.append(",");
		}
		preferenceFile << pString << endl;
	}
```

```
        for (int i = 0; i < numCoalitions; i++) {
                pString.clear();
                for (int j = 0; j < numAgents; j++) {
                        pString.append(to_string(colArray[i][j]));
                        if (j < numAgents - 1)
                                pString.append(",");
                }
                preferenceFile << pString << endl;
        }
        preferenceFile.close();
}

bool CheckCoalitionStructure(int *ptr, int x) {
        bool checked = true;
        int* p = max_element(ptr, ptr + numAgents);             /* max # of coalitions */
        string binStr;
        string binStr2;
        int value;
        int expValue;

        for (int j = 0; j < numAgents; j++) {
                colStructureValue[j] = 0;
        }
        for (int i = 1; i <= *p; i++) {
                expValue = numAgents - 1;
                value = 0;
                binStr.clear();

                /* Identify the various coalitions within the coalition structure*/
                for (int j = 0; j < numAgents; j++) {
                        if (ptr[j] == i) {
                                binStr.append("1");
                                value += pow(2, expValue);
                        }
                        else
                                binStr.append("0");
                        expValue--;
                }
                /* Enter the coalition structure values */
                for (int j = 0; j < numAgents; j++) {
                        if (binStr[j] == '1')
                                colStructureValue[j] = colArray[value][j];
                }
                /* Determine if it satisfies individual rationality */
                if (blockedCoalitionArray[value] == 1) {
                        checked = false;
                        return checked;
                }
        }
        return checked;
}

bool CheckDominance(int *ptr) {
        bool blocked = true;
        string binString;
        for (int i = 1; i < numCoalitions; i++) {
                if (blockedCoalitionArray[i] == 0) {
                        binString = ConvertToBinary(i);
                        blocked = true;
                        for (int j = 0; j < numAgents; j++) {
```

```
                    if (binString[j] == '1') {
                        int x = colStructureValue[j];
                        if (colArray[i][j] <= colStructureValue[j]) {
                            blocked = false;
                            j = numAgents;
                        }
                    }
                }
                if (blocked)
                    return false;
            }
        }
        return true;
}

void FindCoalitionStructures() {

        /* Create output file name */

        //outFile = "C:\\Users\\Daniele\\Dropbox\\Dissertation\\GT Prototype\\Strict Hedonic
Results\\";
        outFile.clear();
        outFile.append("Output");
        outFile.append(fileAppend);
        cout << outFile << endl;

        ofstream outputFile(outFile);
        outputFile.open(outFile, ios::out);
        if (outputFile.is_open())
            outputFile.clear();
```

```
/* Test write file */
if (outputFile.fail())
        cout << "Output file -- fail bit set" << endl;
if (outputFile.eof())
        cout << "eof bit set" << endl;
/**/

int r = 1;
int j = 0;
int n1 = numAgents - 1;
int *b = NULL;
int *c = NULL;
string writeString;

b = new int[numAgents];
c = new int[numAgents];
for (int i = 0; i < numAgents; i++) {
        b[i] = 0;
        c[i] = 0;
}
c[0] = 1;
b[0] = 1;

bool continueLoop = true;
colStructureValue = new int[numAgents];



if (outputFile.is_open()) {
        while (continueLoop) {
```

```
while (r < n1) {
        r++;
        c[r - 1] = 1;
        j++;
        b[j] = r;
}
int x = numAgents - j + 1;
for (int y = 1; y < x; y++) {
        c[n1] = y;
        /* check/write if coalition structure is not blocked */
        if (CheckCoalitionStructure(c, x)) {
                /* determine dominance*/
                if (CheckDominance(c)) {
                        /* write to file */
                        for (int i = 0; i < numAgents; i++) {
                                writeString.clear();
                                writeString.append(to_string(c[i]));
                                writeString.append(",");
                                outputFile << writeString;
                                //outputFile << c[i] << " ";
                                cout << c[i] << ",";
                        }
                        outputFile << endl;
                        cout << endl;
                }
        }
}
r = b[j];
c[r - 1]++;
if (c[r - 1] > r - j)
```

```
                    j--;
                if (r == 1) {
                    continueLoop = false;
                    cout << "End coalition formation loop" << endl;
                }
            }
            outputFile.close();
        }
        else
            cout << outFile << " does not exist";
}

void DetermineBlockedCoalitions() {
        /* Based on a strict hedonic preference, determine which coalitions are blocked */

        /* Create an array of values each agent would have if they were on their own */
        /* Rationale: no agent will accept a coalition that is not at least as good as individuality
(individually rational) */

        int n = numAgents;
        singletonArray = new int[n];
        int singletonRow = 0;
        for (int i = 0; i < n; i++) {
            singletonRow = pow(2, i);
            singletonArray[n - i - 1] = colArray[singletonRow][n - i - 1];
        }

        /* Mark any coalition that defies individual rationality */
        string bString;
        string bString2;
        blockedCoalitionArray = new int[numCoalitions];
        for (int i = 0; i < numCoalitions; i++) {
                bString = ConvertToBinary(i);
                blockedCoalitionArray[i] = 0;
                for (int j = 0; j < n; j++) {
                        /* Block coalition if agent is better off alone (individual rationality) */
                        if (bString[j] == '1' && colArray[i][j] < singletonArray[j]) {
                                blockedCoalitionArray[i] = 1;
                                //cout << "Blocked Coalition: " << i << endl;
                                j = n;
                        }
                }
        }

        /* Compare all coalitions */



        cout << "Ending blocked coalition structures" << endl;
        /* Still need to check for other blocked coalitions*/
}

/*void SetCoalitionValues() {


        inFile = "C:\\Users\\Daniele\\Dropbox\\Dissertation\\GT Prototype\\Strict Hedonic Inputs\\";
        inFile.append(to_string(numAgents));
        inFile.append("AHedonicPreference.txt");
        cout << inFile << endl;

        ifstream inputFile;
```

```
string inputString;
stringstream lineStream;
string c;

/* Test read file
inputFile.open(inFile, ios::in);
if (inputFile.fail())
	cout << "Input file -- fail bit set" << endl;
if (inputFile.eof())
	cout << "eof bit set" << endl;
/**/

/* Dynamic 2D array allocation
colArray = new int *[numCoalitions];
for (int i = 0; i < numCoalitions; i++) {
	colArray[i] = new int[numAgents];
	for (int j = 0; j < numAgents; j++)
		colArray[i][j] = 0;
}

/* Read in a .csv file into a 2d array
if (inputFile.is_open()) {
	for (int i = 0; i < numCoalitions; i++) {
		getline(inputFile, inputString);
		lineStream.clear();
		lineStream.str(inputString);
		int j = 0;
		while (getline(lineStream, c, ',')) {
			colArray[i][j] = stoi(c);
			j++;
```

```
                    }
                }
            }
            else
                cout << inFile << " does not exist";
}*/

int main() {

            srand((unsigned)time(NULL));

            for (int x = 4; x < 16; x++) {
                /*********************************************/
                /*****   INSERT NUMBER OF AGENTS FOR GAME  ****/

                numAgents = x;  /* number of agents in game  */
                numCoalitions = pow(2, numAgents);

                /*********************************************/

                time_t now = time(0);
                struct tm ltm;
                localtime_s(&ltm, &now);

                string pFile;
                pFile.clear();
                pFile.append("Duration");
                fileAppend.clear();
                fileAppend.append(to_string(numAgents));
                fileAppend.append("Agents");
fileAppend.append(to_string(ltm.tm_year));
fileAppend.append(to_string(ltm.tm_mon));
fileAppend.append(to_string(ltm.tm_mday));
fileAppend.append(".txt");
pFile.append(fileAppend);

ofstream durationFile;
durationFile.open(pFile, ios::app);



/* Test write file */
if (durationFile.fail())
        cout << "Duration file -- fail bit set" << endl;
if (durationFile.eof())
        cout << "Duration eof bit set" << endl;

/* 3 primary functions in the driver module     */
/*    Set coalition values                      */
/*    Determine blocked coalitions              */
/*    Find coalition structures within core     */

for (int i = 51; i < 100; i++) {
        // Provide 10 sample files
        fileAppend.clear();
        fileAppend.append(to_string(numAgents));
        fileAppend.append("Agents");
        fileAppend.append(to_string(ltm.tm_year));
        fileAppend.append(to_string(ltm.tm_mon));
        fileAppend.append(to_string(ltm.tm_mday));
        fileAppend.append(to_string(ltm.tm_hour));
```

```
                    fileAppend.append(to_string(ltm.tm_min));
                    fileAppend.append("-");
                    fileAppend.append(to_string(i));
                    fileAppend.append(".txt");

                    //SetCoalitionValues();
                    cout << "Set  Coalition Values" << endl;
                    SetRandomCoalitionValues();
                    auto start = high_resolution_clock::now();
                    cout << "Determine Blocked Coalitions" << endl;
                    DetermineBlockedCoalitions();
                    cout << "Find Coalition Structures" << endl;
                    FindCoalitionStructures();
                    auto stop = high_resolution_clock::now();
                    auto duration = duration_cast<microseconds>(stop - start);

                    durationFile << pFile;
                    durationFile << ",";
                    durationFile << to_string(duration.count());
                    durationFile << endl;
                    cout << pFile << "," << to_string(duration.count()) << endl;
                }
                durationFile.close();
        }

        cout << "End Program" << endl;

        /* REMEMBER TO CLEAR ALL POINTERS! */
}
```

APPENDIX: NETLOGO PROGRAM FOR ABCG MODEL

```
;; **********************************************************
;; ***  COALITION FORMATION ROUTINES
;; **      Use ABM to find a core member
;; ***  D. Vernon-Bido
;; **       Version 6.6    8/7/21
;; ***  Change trio to random size coalition   dvb 7/18
;; ***  Sensitivity analysis  dvb 8/9
;; ***      8/25 starting from solo coalition
;; **   Strict Hedonic Version
;; **********************************************************
;; ** Migrate from Glove Game to Strictly Hedonic game 3/17/20
;; ** Refactor code  5/30/20
;; ** Migrate from pairs to increased combination sizes 5/31/20
;; ** Corrected bug in determine value (binStr was reversed) 6/20/21


extensions [ array matrix ]

globals [

 ;max-agents                ; number of agents in game (N)

 coalitionValueInput       ; string of coalition values from input file

 coalitionValueList        ; list of coalition values for a given coalition

 coalitionValueMatrix      ; 2D array of coalition values

 max-coalitions            ; number of coalitions possible

 decValue                  ; index value of current coalition

 decCounter                ; size of current coalition

 fname                     ; input file name

 next-color

 control-master            ; list of all control #s

 control-number            ; array of control #

 orderMatrix               ; order of routine execution

 agent-list                ; random ordered list of agents

 next?                     ; boolean for checking end of list


 ;; Variables for metrics

 total-coalitions          ; number of existing coalitions
 max-agent-value          ; largest agent coalition value v(S)
 total-coalition-value    ; sum of all coalition values
 grand-coalition-value    ; v(N)
 avg-agent-value           ; average agent payoff
 total-joins
 total-splits
 coalition-list
 prior-coalition-list
 final-coalition-list
 note-tick
 gl-master
]
turtles-own [
 coalitionAdder            ; agent's additive value to determine coalition number   *STATIC*
 singletonValue           ; agent's value in singleton coalition                     *STATIC*
 currentCoalition         ; agent's current coalition
 currentValue             ; agent's value in current coalition
 newCoalition             ; agent's proposed coalition
 newValue                  ; agent's value in proposed coalition
 coalitionSize             ; coalition size
 priorValue               ; agent's previous value
 priorCoalition            ; agent's previous coalition
]

;; SETUP -- Initialize turtles and groups
;;
;; Determine initial groups and coalition values
to setup
 clear-all
 set next-color           5
```

```
;set max-agents           4
set max-agent-value       0
set max-coalitions        (2 ^ max-agents)
set total-coalitions      max-agents

;; Create agents
crt max-agents [
 set color 99
 set shape "person"
 set size 2
 setxy (random-xcor * .85) (random-ycor * .85)
 set newCoalition random max-agents                  ; randomly set next possible coalition
 set coalitionAdder (2 ^ (max-agents - who - 1))     ; determine coalition adder value
 set currentCoalition (2 ^ who) - 1           ; set current coalition number to singleton coalition
 set coalitionSize 1                          ; # of agents in coalition -- coalition size = 1
 set currentValue 0                           ; current value = 0
 set priorValue 0                             ; previous value = 0
 set priorCoalition 0                         ; previous coalition = 0
]

;; *** List of master control numbers – used to load files
set control-master ["0" "0" "0" "0" "120117859" "120117859" "120117859" "12011790" "12011790"
"12011791" "12011793" "12011797" "120117181" "1201171958" "12012076" "1214291512"]
set control-number array:from-list control-master

;; *** Create a matrix containing all the coalition values read in from a text file
set coalitionValueMatrix matrix:make-constant max-coalitions max-agents 0     ; creates a matrix with
dimension # of coalitions x number of agents & initializes to zero

;set fname "…\\Strict Hedonic Inputs\\xAgents\\InputxAgents-y.txt"
```

```
 let cn array:item control-number max-agents
 set fname "...\\Hedonic Inputs\\"
 set fname (word fname "Input" max-agents "Agents" cn "-" selector ".txt")

 ;show fname

 ifelse file-exists? fname [                                    ; check for input text file
  file-close
  file-open fname
  let i 0
  let j 0
  let continue true
  let firstLine true
  let ss1 0
  let pos 0
  while [continue] [
   set coalitionValueInput file-read-line                                    ; read input string
   set j 0
   if empty? coalitionValueInput[
    set j max-agents
    set continue false
   ]
   ;show coalitionValueInput
   while [j < max-agents - 1][
    set pos position "," coalitionValueInput                                    ; find the comma
    set ss1 substring coalitionValueInput 0 pos                                    ; split string at comma
    set coalitionValueInput substring coalitionValueInput (pos + 1) (length coalitionValueInput)      ;
remove value from input string
    matrix:set coalitionValueMatrix i j (read-from-string ss1)                              ; move coalition values
into matrix
    set j j + 1
    ]
    if continue [
      matrix:set coalitionValueMatrix i j (read-from-string coalitionValueInput)
    ]
    set i (i + 1)
  if file-at-end? [
      set continue false
      file-close
    ]
  ]
][
  show "File Not Found!"
]
;show coalitionValueMatrix
;; *** Initialize values
ask turtles [
  set singletonValue matrix:get coalitionValueMatrix   (2 ^ (max-agents - who - 1))  (who)
  set label who
]

let i 0
; ** test next possible coalitions to see if payoff improves
while [i < max-agents][
  if any? turtles with [newCoalition = i][
    check_coalition i
  ]
 set i i + 1
]

; initial conditions
```

```
 ;fully_connect_network
 ask turtles [ set newCoalition  0 ]  ; clear test coalition value
 gather_metrics
 reset-ticks
end
;;****** END SETUP ROUTINE *************
;___________________________________________________________________________________

;; ***** PROGRAM EXECUTION ***************

to go
 if ticks = 50000 [
  convert_coalition_list             ; make coalition list readable
  show final-coalition-list
  show note-tick
  stop
 ]
 set agent-list (list )
 ask turtles [
  set agent-list lput who agent-list
 ]

 set next? true
 foreach agent-list [ [ i ] -> create_random_group i]
 gather_metrics

 if (total-coalitions > 1) [
  join_coalition
 ]
 remove_from_coalition
 pair_coalition
 trio_coalition
 defect_coalition
 split_coalition
 single_coalition
 gather_metrics
 convert_coalition_list
 if (coalition-list != prior-coalition-list)[
  set note-tick ticks
  set prior-coalition-list coalition-list
 ]
 tick
end
;___________________________________________________________________________


;; ** This routine creates a fully connected network as the initial condition
to fully_connect_network
 ask turtles [
  create-links-with other turtles
  set currentCoalition (max-coalitions - 1)
  set currentValue matrix:get coalitionValueMatrix currentCoalition  who
  set coalitionSize max-agents
 ]
 set total-coalitions 1
 repeat 5 [layout-spring turtles links .2 7.5 .5]
end
;___________________________________________________________________________
```

```
;;******* GENERIC ROUTINES TO CHECK FOR UPDATES AND PROVIDE HOUSEKEEPING *********

;; *** Determine newValue
to determine_newValue [ coNo ]
 ;show "enter determine_newValue"
 let binStr array:from-list n-values max-agents [0]
 ask turtles with [ newCoalition = coNo ][
  array:set binStr who 1
 ]

 let expValue (max-agents - 1)
 set decValue 0
 set decCounter 0
 let i 0
 while [expValue >= 0][
  if array:item binStr i = 1 [
   set decValue ( decValue + (2 ^ expValue) )
   set decCounter + 1
  ]
  set expValue (expValue - 1)
  set i + 1
 ]

 ;show matrix:get-row coalitionValueMatrix decValue
 ask turtles with [ newCoalition = coNo ][
  ;set newValue matrix:get coalitionValueMatrix decValue (max-agents - who - 1)
  set newValue matrix:get coalitionValueMatrix decValue who
 ]
end
```

```
;______________________________________________________________________________

; Reset newValue and newCoalition
to reset_newValue_newCoalition
  ask turtles with [newCoalition > 0 ][
    set newValue 0
    set newCoalition 0
  ]
end

;_______________________________________________________________________________

; Set newValue and newCoalition for individual rationality
to rearrange_single [ t1 coNo ]
  let tempCoalitions max-coalitions + 9
  ask turtles with [ currentCoalition = coNo ][
    if t1 != who [ set newCoalition tempCoalitions]
  ]
  exit_coalition t1 tempCoalitions
  reset_newValue_newCoalition
  set_color_and_links  decValue

end
;______________________________________________________________________________

; This routine test the value of a new coalition to see if all the agent's in the coalition
; improve their payoff. If so, the new coalition is accepted and updated.

to check_coalition [ coNo ]
```

```
 let update?   true
 let tempCount 0

 determine_newValue coNo
 ask turtles with [newCoalition = coNo][
  if (newValue <= currentValue) [ set update? false]
 ]

 set tempCount count turtles with [newCoalition = coNo]
 ;ensure that there are turtles with newCoalition available to check
 if tempCount = 0 [
  show "tempCount error"
  set update? false
 ]

 ifelse update? [
  update_coalition coNo
  set next? false
 ][
  set next? true ]
end

to update_coalition [ coNo ]
 ;show "update coalition"
 determine_newValue coNo
 ask turtles with [newCoalition = coNo][
  ;show newCoalition
  ask my-links [die]
  set priorValue         currentValue
  set priorCoalition     currentCoalition
```

```
   set currentValue       newValue
   set currentCoalition    decValue
   set coalitionSize      decCounter
   set newCoalition       0
   set newValue          0
  ]

  set_color_and_links  decValue             ; designate new group with color change and link updates

end
;______________________________________________________________________________

; This routine removes an agent from a coalition and updates the coalition values

to exit_coalition [ splitTurtle splitCoalition ]
  ask turtle splitTurtle [
   ;show "exit coalition"
   set priorValue        currentValue
   set priorCoalition     currentCoalition
   set currentValue       singletonValue
   set currentCoalition   coalitionAdder
   set newCoalition       0
   set newValue          0
   set coalitionSize      1
   set color 99
   ask my-links [die]
  ]
  update_coalition splitCoalition

  reset_newValue_newCoalition
```

```
end
;________________________________________________________________________________

; Designate coalitions with color and links
to set_color_and_links [ coNo ]
  ;show "set color and links"
  ask turtles with [currentCoalition = coNo ][
    set color next-color
      create-links-with other turtles with [currentCoalition = coNo]
  ]
  set next-color next-color + 7
  if next-color > 125 [set next-color 15]
  repeat 5 [layout-spring turtles links .2 7.5 1]
end
;________________________________________________________________________________

;; *************** FUNCTION ROUTINES ***************************************

;; ********** RANDOM COALITION ROUTINE **************
; Create a random subgroup including the current turtle
; Group size is randomly selected

to create_random_group [ this-turtle ]
  ;show "create_random_group"
  reset_newValue_newCoalition
  let tempCoalition max-agents + 8
  ;let newGroup -1
  let j -1
  set next? true
  ask turtle this-turtle [
    set tempCoalition who + max-agents * 10
    set newCoalition tempCoalition
  ]

  ;show "Check Sub-group"
  ; select a random number of agents to create a subgroup with this-turtle
  let groupSize random (max-agents / 2) + 1
  ask n-of groupSize turtles with [newCoalition != tempCoalition]  [
      set newCoalition tempCoalition
  ]

  set groupSize count turtles with [newCoalition = tempCoalition]
  ;set errorStr "Sub-group"
  ;set errorStr (word errorStr " turtle: " this-turtle " size: " groupSize)
  check_coalition tempCoalition

  ; if the subgroup does not provide better imputation remove this-turtle
  if next? and (groupSize > 1)[
    ;set errorStr "Exit-group"
    ;set errorStr (word errorStr " turtle: " this-turtle " size: " groupSize)
    ask turtle this-turtle [ set newCoalition -1]
    check_coalition tempCoalition
  ]

  ; if subgroup does not provide better imputation try merging groups
  if next? [
    ;set errorStr "Merge-group"
    ;set errorStr (word errorStr " turtle: " this-turtle " size: " groupSize)
    ask turtle this-turtle [
      set newCoalition tempCoalition
    ]
    ask one-of turtles with [newCoalition != tempCoalition][
      set j newCoalition
    ]
    ask turtles with [currentCoalition = j][
      set newCoalition tempCoalition
    ]
    check_coalition tempCoalition
  ]
  reset_newValue_newCoalition
end
;; ********** JOIN COALITION ROUTINE **************

; Select two coalitions to possibly merge
to join_coalition
  ;show "join coalition"
  let numberOfGroups random (total-coalitions )
  let tempCoalition max-coalitions + 2  ; ensure that it does not match any coalition
  let coNo max-coalitions + 2
  let sameGroup? true
  let testNo 0
  let testwho -1
  ; randomly pick a turtle and use its coalition for possible merger
  let x 0
  let y 0

  ;if (numberOfGroups < 2 and numberOfGroups > 0) [ set numberOfGroups 2 ]
  let str "# of coalitions = "
  set str (word str total-coalitions)
  ;show str
```

```
 if (numberOfGroups = 1) [ set numberOfGroups 2 ]
 while [x < numberOfGroups][
  set y count turtles with [newCoalition != tempCoalition]
  ask one-of turtles with [newCoalition != tempCoalition][
   set newCoalition tempCoalition
   set coNo currentCoalition
   set testwho who
  ]
  ask turtles with [currentCoalition = coNo][
   set newCoalition tempCoalition
  ]
  set x (x + 1)
 ]
 let update? true
 determine_newValue tempCoalition
 ask turtles with [ newCoalition = tempCoalition][
  if (newValue <= currentValue) [ set update? false]
 ]
 if update? [
  ;show "update join coalition"
  update_coalition tempCoalition
 ]
 reset_newValue_newCoalition
end
;_____________________________________________________________________________

;; ********** REMOVE FROM COALITION ROUTINE **************

; This routine randomly selects an agent to test if the coalition is better off
; removing the agent from the group (kick-out function)
```

```
to remove_from_coalition
 ;show "remove from coalition routine"
 let tempCoalition max-coalitions + 3
 let splitTurtle   -1
 let update? true
 ; first check that there is a coalition of size greater than 1
 if count turtles with [coalitionSize > 1] > 0 [
  ask one-of turtles with [coalitionSize > 1][
   set tempCoalition currentCoalition
   set splitTurtle   who
  ]
  ask turtles with [currentCoalition = tempCoalition][
   if who != splitTurtle [
    set newCoalition tempCoalition
   ]
  ]
  determine_newValue tempCoalition
  ask turtles with [ newCoalition = tempCoalition ][
   if newValue <= currentValue [ set update? false ]
 ]
  if update? [
   exit_coalition splitTurtle tempCoalition ]
  reset_newValue_newCoalition
 ]
end
;____________________________________________________________________________
;; *********** TRIO COALITION ROUTINE ************


; This routine randomly select three agents to create a new coalition.
; If the all agents of the pair are better off, each departs from current coalition
; to form a new coalition

to trio_coalition
 ;show "trio coalition"
 let x 0
 let coSize random (max-agents - 2)
 let coalitionList []
 let coalList []
 let tempCoalition max-coalitions + 4
 let ncTemp max-agents + 14
 let update? true

 ; select the number of agents for the new coalition (between 2 and max-agents)
 set coSize + 2
 ; randomly choose agents and place in a list
 while [x < coSize][
  set coalitionList lput (random max-agents) coalitionList
  set x x + 1
 ]
 set coalitionList remove-duplicates coalitionList         ;remove duplicates from the list
 ; cycle thru each agent listed and set newCoalition for testing
 foreach coalitionList [
  a -> ask turtle a [
   set newCoalition tempCoalition
   set coalList lput currentCoalition coalList
  ]
 ]
 ; determine if the new coalition is better
 determine_newValue tempCoalition
```

```
  ask turtles with [newCoalition = tempCoalition][
    if newValue <= currentValue [ set update? false ]
  ]

  if update?[
    update_coalition tempCoalition                          ; update new coalition
    ; update coalitions that had agents removed to reflect new coalition
    set coalList remove-duplicates coalList
    foreach coalList [
      a -> ask turtles with [currentCoalition = a][
        set newCoalition ncTemp
      ]
      update_coalition ncTemp
      set ncTemp ncTemp + 10
    ]

  ]
  reset_newValue_newCoalition
end
;________________________________________________________________________________
;; ********** PAIR COALITION ROUTINE **************

; This routine randomly selects two agents to create a new coalition.
; If the both agents of the pair are better off, each departs from current coalition
; to form a new coalition



to pair_coalition
  ;show "pair coalition routine"
  let t1 -1
let t2 -1
let tempCoalition max-coalitions + 5
let oldCoalitionT1 -1
let oldCoalitionT2 -1
let update? true
ask one-of turtles [
 set t1 who
 set newCoalition tempCoalition
 set oldCoalitionT1 currentCoalition
]
ask one-of turtles with [who != t1][
 set t2 who
 set newCoalition tempCoalition
 set oldCoalitionT2 currentCoalition
]
determine_newValue tempCoalition
ask turtle t1 [
 if newValue <= currentValue [ set update? false ]
]
ask turtle t2 [
 if newValue <= currentValue [ set update? false ]
]
if update?[

 let splitCoalition 0
 let coSize 0
 let nCtemp max-agents + 15
 let nCtemp2 max-agents + 25

 update_coalition tempCoalition          ; update pair to new coalition
```

```
    ask turtles with [currentCoalition = oldCoalitionT1][
      set newCoalition nCtemp
    ]

    ifelse (oldCoalitionT1 = oldCoalitionT2)[
      ask turtles with [currentCoalition = oldCoalitionT2][
        set newCoalition nCtemp
      ]
      update_coalition nCtemp
    ][
      ask turtles with [currentCoalition = oldCoalitionT2][
        set newCoalition nCtemp2
      ]
      update_coalition nCtemp
      update_coalition nCtemp2
    ]
  ]
  reset_newValue_newCoalition
end
;___________________________________________________________________________________

;; ********** DEFECT FROM COALITION ROUTINE **************

; This routines randomly selects an agent to test whether to join a different, randomly selected
; coalition
to defect_coalition
  ;show "defect coalition routine"
  let t1 -1
  let t1Coalition 0
let t2Coalition 0
let tempCoalition max-coalitions + 6
let update? false
reset_newValue_newCoalition

; select an agent
ask one-of turtles [
 set t1 who
 set t1Coalition currentCoalition
 set newCoalition tempCoalition
]
; select an alternate coalition to join
if count turtles with [currentCoalition != t1Coalition] > 0 [
 ask one-of turtles with [currentCoalition != t1Coalition][
   set t2Coalition currentCoalition
   set newCoalition tempCoalition
 ]
 ; check if new coalition is improves value for members
 determine_newValue tempCoalition
 ask turtles with [ currentCoalition = t2Coalition ][
   if newValue <= currentValue [ set update? false ]
 ]
 ask turtle t1 [
   if newValue <= currentValue [ set update? false ]
 ]


 if update? [
   update_coalition tempCoalition
   ask turtles with [ currentCoalition = t1Coalition ] [ set newCoalition t1Coalition ]
```

```
      update_coalition t1Coalition
    ]
  ]
  reset_newValue_newCoalition
end


;; ********** SPLIT COALITION ROUTINE **************

; This routine breaks a group into subgroups and examines if either subgroup is better off
to split_coalition
  ;show "split coalition routine"
  let continue? true
  let counter 0
  let tempCoalition  max-coalitions + 7
  let temp2Coalition max-coalitions + 17
  let t1Coalition 0
  let tempSize 0
  let g1 true
  let g2 true

  ; select a coalition of size > 2
  if count turtles with [ coalitionSize > 2 ] > 0 [
    ask one-of turtles with [ coalitionSize > 2 ][
      set t1Coalition currentCoalition
      set tempSize coalitionSize
    ]

    ; split the group using a uniform random number
    ; randomly select turtles for group 1 until the number
  ; generated is greater than 0.5 all others are in group 2

  ask turtles with [currentCoalition = t1Coalition ][
    ifelse (continue?) and (random-float 1 < 0.5) [
      set newCoalition tempCoalition
      set counter + 1
    ][
      set continue? false
    ]
  ]



  ; if the split is greater than 0 but smaller than the original coalition size
  ; check the split for coalition improvement
]
; did the group split into 2?
if (counter > 0) and (tempSize - counter > 0) [
  ;show "split group"
  if counter <= 0 [show "counter error"]
  if tempSize - counter <= 0 [show "tempSize error"]
  ask turtles with [ currentCoalition = t1Coalition][
    if newCoalition != tempCoalition [ set newCoalition temp2Coalition ]
  ]
  ;determine whether the coalition split is beneficial to either group
  determine_newValue tempCoalition
  ask turtles with [newCoalition = tempCoalition][
    if newValue <= currentValue [ set g1 false ]
  ]
  determine_newValue temp2Coalition
  ask turtles with [newCoalition = temp2Coalition][
    if newValue <= currentValue [ set g2 false ]
   ]

   ; if there is a benefit, split the groups into two coalitions
   if (g1) or (g2) [
    ;show "update split coalition"
    ask turtles with [newCoalition = tempCoalition][
     ask my-links [ die ]
     ;show "group 1 turtle"
    ]
    ;set_color_and_links tempCoalition
    update_coalition tempCoalition
    ;show "Group 2"
    ask turtles with [currentCoalition = temp2Coalition][
     ask my-links [ die ]
     ;show "group 2 turtle"
    ]
    ;set_color_and_links temp2Coalition
    update_coalition temp2Coalition
   ]

   reset_newValue_newCoalition
  ]
end


;; ********** INDIVIDUAL RATIONALITY ROUTINE **************

; This routines tests for individual rationality. All agents check the payoff in their current coalition
; to their individual value. If the coalition value is lower, they exit the coalition
```

```
to single_coalition
  ;show "single coalition routine"
  ask turtles with [coalitionSize > 1][
    if singletonValue > currentValue [
      rearrange_single who currentCoalition
    ]
  ]
end

;; ************************************ METRIC FUNCTIONS
****************************************


; This routine gathers metrics for evaluation
to gather_metrics
  let avg 0
  let cv 0
  let tcv 0
  let tc 0
  set coalition-list []
  let i 0
  while [i < max-agents][
    ask turtle i[
      set tcv tcv + currentValue
      if currentValue > max-agent-value [set max-agent-value currentValue]     ; max imputation
      if member? currentCoalition coalition-list = false [
        set tc tc + 1
      ]
      set coalition-list lput currentCoalition coalition-list
    ]
    set i i + 1
  ]
  set total-coalitions tc
  set total-coalition-value tcv
  set avg-agent-value total-coalition-value / max-agents
end


; Make the coalition list readable
to convert_coalition_list
  set final-coalition-list []
  let temp-coalition-list []
  let i 0
  let nextPosition 0
  while [i < max-agents][
    ifelse member? (item i coalition-list) temp-coalition-list [
      set final-coalition-list lput (item (position (item i coalition-list) coalition-list) final-coalition-list) final-
coalition-list
    ][
      set temp-coalition-list lput (item i coalition-list) temp-coalition-list
      set final-coalition-list lput nextPosition final-coalition-list
      set nextPosition nextPosition + 1
    ]
    set i i + 1
  ]
  ;show final-coalition-list
end
```

VITA

Daniele M. Vernon-Bido

DEGREES:

Doctor of Philosophy (Computational Modeling and Simulation Engineering), Old Dominion University, Norfolk, VA, May 2022.

Masters (Modeling, Simulation, and Visualization Engineering), Old Dominion University, Norfolk, VA, December 2013.

Master of Science (Computer Information Systems), Boston University, Boston MA, May 1991.

Bachelor of Science (Computer Science), Brooklyn College, Brooklyn, NY, January 1987.

PROFESSIONAL EXPERIENCE:

Navy Continuous Training Environment, NSWC Corona, Norfolk, VA

Scientist	Feb. 2019 – present

SCHOLARLY PUBLICATIONS:

Collins, A., Petty, M., Vernon-Bido, D., & Sherfey, S. (2015). A call to arms: Standards for agent-based modeling and simulation. *Journal of Artificial Societies and Social Simulation*, *18*(3), 12.

Kavak, H., Vernon-Bido, D., & Padilla, J. J. (2018, July). Fine-scale prediction of people's home location using social media footprints. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation* (pp. 183-189). Springer, Cham.

Kavak, H., Padilla, J. J., Vernon-Bido, D., Gore, R., & Diallo, S. (2016, April). A characterization of cybersecurity simulation scenarios. In *SpringSim (CNS)* (p. 3).

Vernon-Bido, D., & Collins, A. J. (2020). Finding Core Members of Cooperative Games Using Agent-Based Modeling. *arXiv preprint arXiv:2009.00519*.

Kavak, H., Padilla, J. J., Vernon-Bido, D., Diallo, S. Y., Gore, R., & Shetty, S. (2021). Simulation for cybersecurity: state of the art and future directions. *Journal of Cybersecurity*, *7*(1), tyab005.

Vernon-Bido, D., Collins, A., & Sokolowski, J. (2015, April). Effective visualization in modeling & simulation. In *Proceedings of the 48th Annual Simulation Symposium* (pp. 33-40).

Lynch, C. J., Kavak, H., Gore, R., & Vernon-Bido, D. (2019). Identifying unexpected behaviors of agent-based models through spatial plots and heat maps. In *Complex Adaptive Systems* (pp. 129-142). Springer, Cham.