Searching for optimal coalition structures

Sandip Sen and Partha Sarathi Dutta
Department of Mathematical & Computer Sciences
University of Tulsa
e-mail:{sandip,partha}@euler.mcs.utulsa.edu

Abstract

Coalition formation has been a very active area of research in multiagent systems. Most of this research has concentrated on decentralized procedures that allow self-interested agents to negotiate the formation of coalitions and division of coalition payoffs. A different line of research has addressed the problem of finding the optimal division of agents into coalitions such that the sum total of the the payoffs to all the coalitions is maximized [4, 8]. This is the optimal coalition structure identification problem. Deterministic search algorithms have been proposed and evaluated under the assumption that the performance of a coalition is independent of other coalitions. We use an order-based genetic algorithm (OBGA) as a stochastic search process to identify the optimal coalition structure. We compare the performance of the OBGA with a representative deterministic algorithm presented in literature. Though the OBGA has no performance guarantees, it is found to dominate the deterministic algorithm in a significant number of problem settings. Additional advantage of the OBGA is its scalability to larger problem sizes and to problems where performance of a coalition depends on other coalitions in the environment.

1 Introduction

A key area of interest to multiagent researchers is the formation of groups by self-interested agents. Agents may want to "join hands" to take advantage of complementary capabilities, resources, and expertise. From an individual agent's point of view the incentive for joining a coalition is to increase the payoff that it can receive. As such, the coalition formation process has received considerable attention in the multiagent systems research [3, 6, 9, 11].

From the overall system designers point of view, however, the total payoff received by all coalitions in the system may be of interest. This total payoff, for example, may be correlated to the level of utilization of system resources. In such cases, the goal would be to maximize the total payoff received by all coalitions which effectively translates to maximum resource utilization. The system designer then would be interested in knowing what collection of coalitions can generate this maximum payoff. A coalition structure (CS) is defined as a partition of the agents in a system into disjoint coalitions. The goal of the system designer is then to find the optimal coalition structure, i.e., the coalition structure that can generate the maximal payoff.

Just because a coalition structure can generate maximal payoff, or alternately, produce optimal resource utilization, does not mean that this will be realized in practice. Also, knowing the optimal coalition structure does not solve the problem of how the agents can be induced to align themselves as per the coalitions contained therein. However, knowing the optimal coalition structure, or some of the best performing coalition structures, can enable the system designer to providing incentives for agents to group into desirable configurations. In any case, knowledge of the optimal coalition structure will allow the system designer to evaluate the relative effectiveness of the current coalition structure in the system.

For reasons cited above, and others, some multiagent researchers have developed and evaluated the performance of anytime algorithms to search for optimal coalition structures in characteristic function games (CFGs) [4, 7, 8]. In CFGs the value of each coalition is given by a characteristic function and the value of a coalition structure is simply defined as the sum of the values of the coalitions that it contains. The deterministic algorithms developed for searching for optimal coalition structures are geared towards systematically searching the space of CSs to provide a bound on the value of the optimal CS [8]. There are two basic shortcomings of these algorithms:

1. These algorithms rely on the assumption that the value of a coalition is independent of other coalitions in the coalition structure. This is justifiable in CFGs but may not be valid in a large number of practical scenarios. In general, how well a group of agents perform is also dependent on other agent groups in the environment.

2. The algorithms exhaustively search the space of all possible coalition structures. Since the size of the space is exponential in the number of agents, these algorithms cannot be used for practical problems. In addition, to even guarantee a bound on the optimal CS payoff, it is necessary to evaluate an exponential number of CSs. Hence, these algorithms are not very useful either to find the optimal CS or to develop a bound on its value for all but the smallest problems.

We propose the use of a stochastic search algorithm, namely an order-based genetic algorithm to search the space of possible coalition structures for CSs with high payoffs. The advantages of the OBGA approach are the following:

- It makes no assumption about how a CS payoff is calculated, i.e., it can be used both for CFGs and non-CFGs.
- It scales up well with increase in the number of agents and hence the size of the space of coalition structures.

The major shortcoming of the OBGA approach is that it provides no guarantees about finding the optimal CS, and that even though it is an anytime algorithm, it cannot specify any bounds on the value of the optimal CS.

Given this contrasting strengths and limitations, we have to arrive at a common, justifiable metric for comparing the performance of the deterministic algorithms listed in Sandholm *et al.* [8] and OBGA algorithms on the optimal CS searching problem. The metric we propose is the best coalition structure found by the algorithms after evaluating a given number of coalition structures. This metric is justifiable as given an allocation of computational resources, e.g., CPU time, we can run both algorithms and observe the best CS found by them. As both algorithms are anytime algorithms, they can also be interrupted at any time to observe the best solution found so far. This periodic interruptions can provide us with snapshots of how the quality of the CSs generated are improving as a function of search effort. .

2 Searching for coalition structures

A coalition structure consists of all the agents in the environment grouped into one or more coalitions. The goal of optimal CS searching algorithms is to find the CS from the space of all CSs, M, that has the maximum value (this can be viewed as the payoff this CS can possibly generate). In characteristic function games the value of a coalition structure, CS, is given by

$$V(CS) = \sum_{S \in CS} v_S,$$

where v_S is the value of the coalition S. In CFGs the value of a coalition is assumed to be determined only by the composition of the coalition and is independent of other coalitions. In general, we will also be interested in situations where the value of a coalition depends on the other coalitions. We will then assume only the presence of a function $V : M \to \mathcal{R}$, that returns the payoff of any coalition structure. The search algorithms attempt to find the optimal coalition structure

$$CS^* = \arg\max_{CS \in M} V(CS).$$

The space M can be viewed as a partially ordered lattice where the supremum element is a CS where every coalition consists of a single element, and the infimum element is the CS with one grand coalition containing all agents. A CS X in the lattice is directly above another CS Y if X can be obtained from Y by splitting one of its coalitions into two coalitions. Thus X has exactly one more coalition than Y. This partial order imposes a structure on the lattice such that we can describe the lattice to have n levels, where n is the number of agents. Every CS at the ith level contains exactly i coalitions. Figure 1 shows such a lattice for 4 agents.

2.1 Deterministic CS search algorithm

The deterministic CS search algorithm SPLIT [7] searches breadth first from the bottom of the lattice. This algorithm obviously does well if the grand coalition (a CS with a single coalition of all agents) is the optimal CS. The performance of this algorithm, as measured by the number of evaluations required before finding the optimal CS, degrades as the level of the optimal CS increases. For CFGs, SPLIT establishes a bound of n on the optimal CS (i.e., finds a CS which is not more than n times worse than CS^*, the optimal coalition structure), after searching 2^{n-1} nodes (all nodes in the lowest two levels of the lattice). No such bounds can be specified by SPLIT for non-CFGs. Though we have experimented only with SPLIT, the other two search algorithms have corresponding strengths and weaknesses and the comparisons presented in this paper holds for these algorithms with minor modifications.

2.2 OBGA CS search algorithm

Genetic algorithms (GAs) are a class of stochastic search algorithms that have been used widely in function optimization [5]. GAs have been particularly effective in large-scale NP-complete combinatorial optimization problems including a wide variety of scheduling and routing problems [1]. The optimal CS search problem is a combinatorial optimization problem with an exponential search space. As long as there is some regularity in the search space, i.e.,

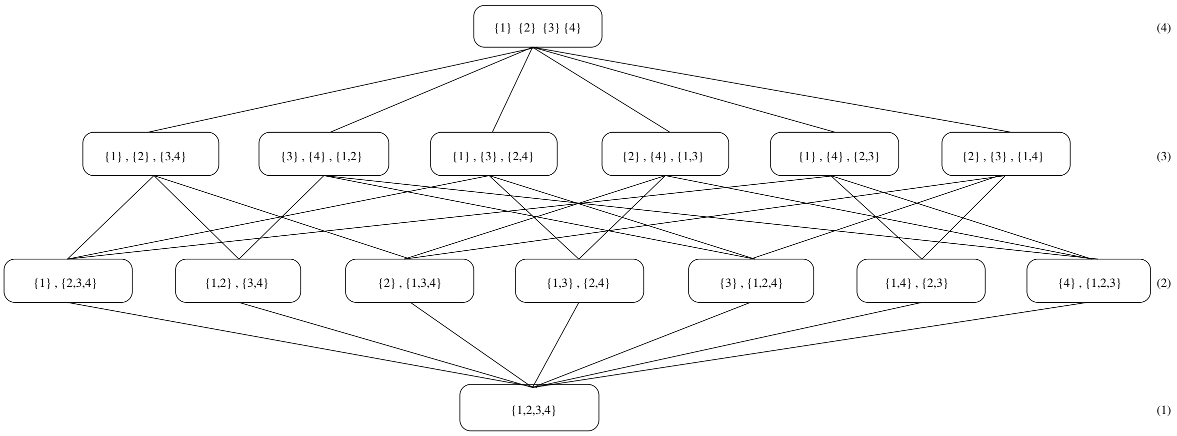

Figure 1. Coalition structure lattice for 4 agents.

the V function is not arbitrary, the GAs have a potential to detect that regularity and hence find CSs that perform relatively effectively. As mentioned before, GAs cannot provide any guarantees of finding the optimal CS or bounding its value with a given number of evaluations. The search performed by the GAs on the lattice is determined by the initial randomly generated population of CSs and the V function. The search is likely to span several levels at the same time, and the next CSs to be evaluated are produced based on the current set of CSs in the GA population.

A simple GA consists of evaluation, selection, and recombination routines, where evaluation evaluates current population members (a CS in our case), selection selects members repeatedly with replacement and based on their evaluation to generate a new population, and recombination constructs new members from the selected members by exchanging and modifying their contents [2].

We next discuss the representation used in our OBGA. We number the n agents from $0, \dots, n-1$. We also include $n-1$ markers numbered from $n, \dots, 2n-2$. The markers are required to separate the agents into different coalitions and since the supremum CS contains n coalitions, we will need $n-1$ markers to represent it. It would have been more economical to use a single symbol to represent markers. We adopted this larger cardinality representation to use a constant length GA representation and for the ease of working with the GALib package (`http://lancet.mit.edu/ga/`). Figure 2 shows example GA structures and corresponding CSs. It is instructive to note that many GA structures map into the same CS (given a coalition structure CS, at least $(n-1)!|CS|!\prod_{S\in CS}|S|!$ GA structures map into one CS). While this representation significantly inflates the GA search space, such many-to-one mappings alleviate the problem somewhat and guards against disruptive crossover by providing non-coding segments or introns [5] (for example crossover of segments including only marker symbols will not disrupt any building blocks).

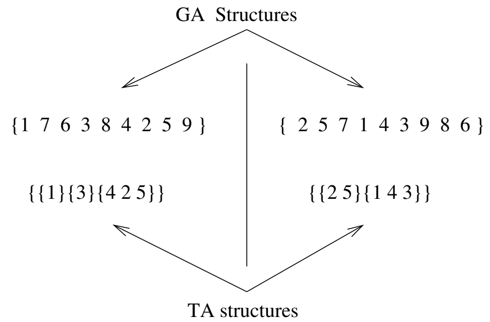

Figure 2. GA population members and the corresponding coalition structures.

In a simple GA the prevalent crossover operators used such as single, two-point, and uniform crossovers performs a position based exchange of structural components between the two selected parents. Such an exchange will produce unacceptable CSs as shown in Figure 3. We need crossover and mutation operators that guarantees that ev-

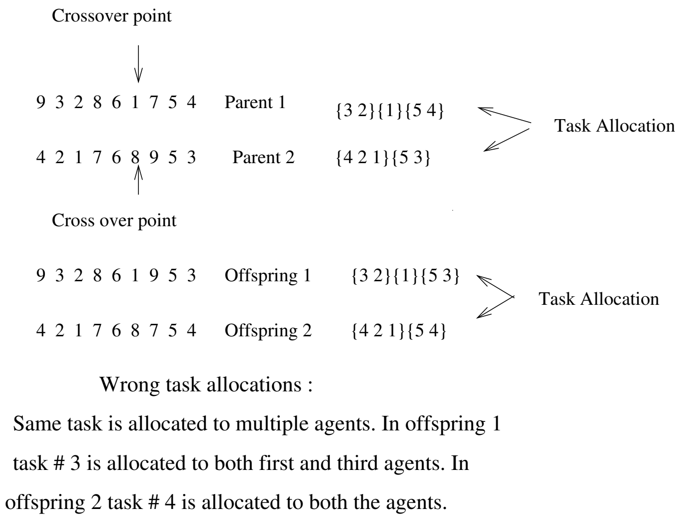

Figure 3. Problem of simple two-point crossover with coalition structures.

ery agent is represented exactly once in each individual CS. Of the operators available in the GA literature that provides such a guarantee, we used the Edge Recombination Operator [10]. This operator belongs to the class of order-preserving crossover operators and is used primarily in combinatorial optimization problems like the traveling salesman problem. Though order-preservation is not a requirement in the CS search problem, the edge recombination operator fulfills our requirement of producing structures which contain all the agents without replication.

3 Experiments

We now report the results from our comparative evaluation of the OBGA and SPLIT algorithms on randomly generated optimal CS search problems. We defined a suite of problems by analyzing the CS search space. Let us reconsider Figure 1. The optimal solution can reside at any of the n levels. We constructed parameterized V functions such that we can generate problems with the optimal solution residing at our chosen level. The data we will be presenting in this section was generated by experimenting with two V-functions in particular. In one of these, V_1, the value of a coalition was independent of the other coalitions. In the other, V_2, this was not the case. We now present each of these functions (we assume that the agents are numbered 1 through n):

$$V_1(CS) = C * \sum_{S \in CS} (weight(|S|) + \sum_{a \in S} (a - min_b S), \quad (1)$$

where

$$\begin{aligned} weight(x) &= x^2, \text{if } 0 < x < K, \\ &= (x - 2K)^2, \text{if } K \leq x \leq 2 * K, \\ &= 0, \text{ otherwise} \end{aligned}$$

In the above, K is the desired size of each coalition in the optimal CS that we can vary to place the optimal CS at different levels. In Equation 1 the second term inside the summation causes the optimal CS to consist of consecutively numbered agents. For example, with 10 agents and K set to 2, $CS^* = \{\{1\ 2\ 3\ 4\ 5\}\{6\ 7\ 8\ 9\ 10\}\}$.

$$V_2(CS) = C * \sum_{S \in CS} (weight(|S|) + distance(S, CS - S)), \quad (2)$$

where

$$distance(S, C) = \min_{X \in C} |\sum_{a \in S} a - \sum_{b \in X} b|.$$

Equation 2 includes a term where the value of a coalition becomes dependent on the membership of the other coalitions.

In either case, the OBGA has no *a priori* notion of the actual function it is optimizing. Neither does it make any assumptions about the class of functions it is working with. We chose these particular functions to have some regularity in the function space that can be observed in practice. For example, in real problems often groups of a certain size are more stable, flexible, and responsive to be effective in the marketplace.

We first experimented with 10 agents and varying desired group size from 2 (optimal CS at level 5 of the lattice) to 5 (optimal CS at level 2 of the lattice). OBGA parameters included a population size of 50, crossover rate of 0.8, mutation rate of 0.1, and number of generations = 1000. The summary results of running OBGA and SPLIT are presented in Table 1. Quite clearly, as the level number of the optimal increases, SPLIT takes exponentially more evaluations to find the optimal structures. The OBGA results are averaged over 5 random population initializations and there appears to be no significant correlation between the level of optimal CS and the time taken to find it. The performance of the algorithms differ modestly from dependent to independent coalitions. We should still remember that with dependent coalitions, SPLIT loses the bound guarantees it can provide after searching the bottom two levels in the independent coalitions case.

The performance of the OBGA did not vary much for the different desired coalition size (except for the group size of 4). This can be explained by the fact that initial random samplings by the GA population allows the identification of regular patterns. For example, given a desired groups size of n, CSs containing much larger or much smaller groups would get poor evaluation. This allows the GA to quickly

Desired group size	Independent coalitions		Coalitions with dependencies	
	SPLIT	OBGA	SPLIT	OBGA
2	908200	3500	908200	3500
3	60000	2500	122500	3500
4	1300	14500	4000	9500
5	100	5000	100	11000

Table 1. Average number of evaluations taken by SPLIT and OBGA to find the optimal CSs for different optimal coalition sizes. Number of agents is 10. Results are presented for both when the value of a coalition depend on other coalitions and when it does not.

identify groups of the right size. More time is spent on finding the correct group compositions once CSs with the right group sizes are identified.

In addition, it was interesting to note that there was little difference in performance of the OBGA between the function V_1 (corresponds to a CFG) and V_2 (a non-CFG). As mentioned above the OBGA is oblivious to the CFG requirement. It should be mentioned that the OBGA is expected to perform equivalently on other function classes as long as the evaluation function chosen is not arbitrary or particularly ill-behaved, e.g., a needle-in-a-haystack function.

We also experimented with larger problem sizes with OBGA by increasing the number of agents. We experimented with up to 50 agents. Larger problem sizes could not be run with the SPLIT algorithm because of exponentially increasing computational cost. Results from these experiments are presented in Figures 4 and 5. It is clear that only a modest increase in computational resources is required by the OBGA to handle this larger problem scenarios. Optimal solutions are found within about 5000 evaluations in most cases. This is particularly encouraging because we can now attempt to solve the optimal coalition structure recognition problem for realistic problem sizes.

4 Conclusions

In this paper we have presented a stochastic search approach for searching optimal coalition structures. We identify some of the limitations of deterministic search algorithms reported in literature. These include assumption of independence of coalition values and the exponential growth in computational requirements. Stochastic search algorithms like the OBGA algorithm we have adapted, however, provide no performance guarantees or bound on the optimal coalition value. We argue that given a fixed allocation of computational resources we would also be keenly interested in finding the best coalition structure. In this regards, the stochastic algorithms have a distinct edge on the deterministic exhaustive algorithms. The OBGA algorithm, for example, can track any pattern or regularity in the CS evaluation functions and quickly identify good coalition structures.

In the experiments, for a large number of problem scenarios (except when the optimal CS was present in the bottom two levels of the lattice of CSs), the OBGA algorithm took an order of magnitude less computational effort to identify the optimal CS. In addition, the scale up of computational effort was found to be linear in the input size, i.e., the number of agents.

We plan to compare OBGA with heuristic search algorithms like greedy search or best-first search to search the lattice structure. Like OBGA, these algorithms would not be able to provide any bounds, but may scale up well with problem size.

We also plan to investigate the performance of OBGA with a number of different value functions with different regularities and shapes. This will help us understand the strengths and weaknesses of the OBGA algorithm in finding the optimal coalition structure.

We believe that other stochastic algorithms like simulated annealing will also fare competitively with the OBGA on the optimal coalition structure recognition problem. These algorithms bias their search of the space based on limited sampling, and can escape local minima in the search space because of their stochastic nature.

Acknowledgments This work has been supported in part by an NSF CAREER award IIS-9702672. We thank Matthew Wall for making the GALib package available.

References

[1] D. Dasgupta and Z. Michalewicz, editors. *Evolutionary Algorithms in Engineering Applications*. Spring-Verlag, New York, NY, 1997.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

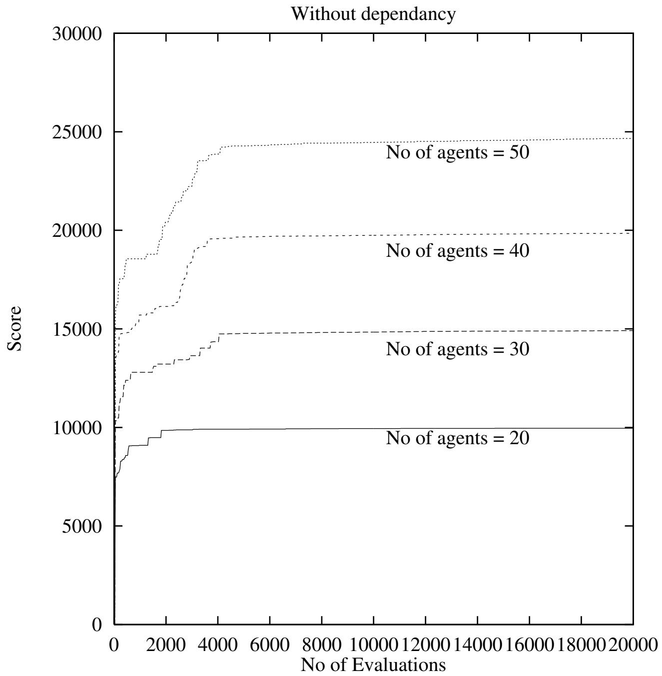

Figure 4. GA performance when coalitions are independent (desired group size is 5).

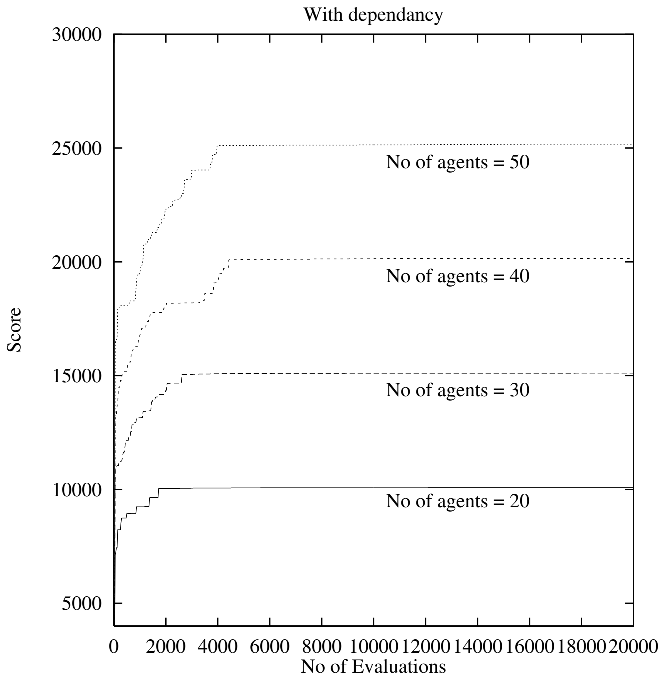

Figure 5. GA performance when coalitions are dependent on each other (desired group size is 5).

[3] S. Ketchpel. Forming coalitions in the face of uncertain rewards. In *Twelfth National Conference on Artificial Intelligence*, pages 414–419. MIT Press/AAAI Press, 1994.

[4] K. Larson and T. W. Sandholm. Anytime coalition structure generation: An average case study. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 40–47, New York: NY, 1999. ACM Press.

[5] M. Mitchell, editor. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[6] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997.

[7] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Anytime coalition structure generation with worst case guarantees. In *Proceedings of the Fifteent National Conference on Artificial Intelligence*, pages 46–53, Menlo Park, CA, 1998. AAAI Press/MIT Press.

[8] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[9] O. Shehory and S. Kraus. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proceedings of the Thirteent National Conference on Artificial Intelligence*, pages 133–140, August 1996.

[10] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, San Mateo, CA, 1991. Morgan Kaufman.

[11] G. Zlotkin and J. S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Twelfth National Conference on Artificial Intelligence*, pages 432–437. MIT Press/AAAI Press, 1994.