Coalition Structure Generation with GRASP

Nicola Di Mauro, Teresa M.A. Basile, Stefano Ferilli, and Floriana Esposito

University of Bari "Aldo Moro", Italy,
{ndm,basile,ferilli,esposito}@di.uniba.it

Abstract. The coalition structure generation problem represents an active research area in multi-agent systems. A coalition structure is defined as a partition of the agents involved in a system into disjoint coalitions. The problem of finding the optimal coalition structure is NP-complete. In order to find the optimal solution in a combinatorial optimization problem it is theoretically possible to enumerate the solutions and evaluate each. But this approach is infeasible since the number of solutions often grows exponentially with the size of the problem. In this paper we present a greedy adaptive search procedure (GRASP) to efficiently search the space of coalition structures in order to find an optimal one.

1 Introduction

An active area of research in multi-agent systems (MASs) is the coalition structure generation (CSG) of agents (equivalent to the complete set partitioning problem). In particular it is interesting to find coalition structures maximizing the sum of the values of the coalitions, that represent the maximum payoff the agents belonging to the coalition can jointly receive by cooperating. A coalition structure is defined as a partition of the agents involved in a system into disjoint coalitions. The problem of finding the optimal coalition structure is $\mathcal{NP}$-complete [1, 2]. Coalition generation shares a similar structure with a number of common problems in theoretical computer science, such as in combinatorial auctions [3], in which bidders can place bids on combinations of items; in job shop scheduling; and as in set partitioning and set covering problems.

Sometimes in MASs there is a time limit for finding a solution, the agents must be reactive and they should act as fast as possible. Hence for the specific task of CSG it is necessary to have approximation algorithms able to quickly find solutions that are within a specific factor of an optimal solution. Hence, the goal of this paper is to propose a new algorithm for the CSG problem able to quickly find a near optimal solution.

The problem of CSG has been studied in the context of characteristic function games (CFGs) in which the value of each coalition is given by a characteristic function, and the values of a coalition structure are obtained by summing the value of the contained coalitions. The problem of coalition structure generation is $\mathcal{NP}$-hard, indeed as proved in [2], given n the number of agents, the number of possible coalition structures than can be generated is $O(n^n)$ and $\omega(n^{n/2})$. Moreover, in order to establish any bound from the optimal, any algorithm must search at least $2^n - 1$ coalition structures. The CSG process can be viewed as being composed of three activities [2]: a) *coalition structure generation*, corresponding to the process of generating coalitions such that agents within each coalition coordinate their activities, but agents do not coordinate between coalitions. This means partitioning the set of agents into exhaustive and disjoint coalitions. This partition is called a coalition structure (CS); b) *optimization*: solving the optimization problem of each coalition. This means pooling the tasks and resources of the agents in the coalition, and solving this joint problem; and c) *payoff distribution*: dividing the value of the generated solution among agents.

Even if these activities are independent of each other, they have some interactions. For example, the coalition that an agent wants to join depends on the portion of the value that the agent would be allocated in each potential coalition. This paper focuses on the coalition structure generation in settings where there are too many coalition structures to enumerate and evaluate due to costly or bounded computation and limited time. Instead, agents have to select a subset of coalition structures on which to focus their search.

Specifically, in this work we adopted a stochastic local search procedure [4], named GRASP [5], to solve the problem of coalition structure generation in CFGs. The main advantage of using a stochastic local search is to avoid exploring an exponential number of coalition structures providing a near optimal solution. Indeed, our algorithm does not provide guarantees about finding the global optimal solution. In particular the questions we would like to pose are:

– **Q1**) can the metaheuristic GRASP be used as a valuable anytime solution for the CSG problem? In many cases, as in CSG, it is necessary to terminate the algorithm prior to completion due to time limits and to reactivity requirements. In this situation, it is possible to adopt anytime algorithms (i.e. algorithms that may be terminated prior to completion and returning an approximation of the correct answer) whose quality depends on the amount of computation.
– **Q2**) can the metaheuristic GRASP be adopted for the CSG problem to find optimal solution faster than the state of the art exact algorithms for CSG problem? In case of optimization combinatorial problems, stochastic local search algorithms have been proved to be very efficient in finding near optimal solution [4]. In many cases, they outperformed the deterministic algorithms in computing the optimal solution.

2 Definitions

Given a set $A = \{a_1, a_2, \ldots, a_n\}$ of n agents, $|A| = n$, called the *grand coalition*, a *coalition* S is a non-empty subset of the set A, $\emptyset \neq S \subseteq A$. A *coalition structure* (CS) $C = \{C_1, C_2, \ldots, C_k\} \subseteq 2^A$ is a partition of the set A, and k is its size, i.e. $\forall i, j$: $C_i \cap C_j = \emptyset$ and $\cup_{i=1}^{k} C_i = A$. For $C = \{C_1, C_2, \ldots, C_k\}$, we define $\cup C \triangleq \cup_{i=1}^{k} C_i$. We will denote the set of all coalition structures of A as $\mathcal{M}(A)$.

As in common practice [2, 6], we consider coalition structure generation in *characteristic function games* (CFGs). In CFGs the value of a coalition structure C is given by $V(C) = \sum_{S \in C} v(S)$, where $v : 2^A \to \mathbb{R}$ and $v(S)$ is the value of the coalition S. Intuitively, $v(S)$ represents the maximum payoff the members of S can jointly receive by cooperating. As in [2], we assume that $v(S) \geq 0$. In case of negative values, it is possible to normalize the coalition values, obtaining a game strategically equivalent to the original game [7], by subtracting a lower bound value from all coalition values. Given a set of agents A, the goal is to maximize the social welfare of the agents by finding a coalition structure $C^* = \arg\max_{C\in\mathcal{M}(A)} V(C)$.

Given n agents, the size of the input to a CSG algorithm is exponential, since it contains the values $v(\cdot)$ associated to each of the $(2^n - 1)$ possible coalitions. Furthermore, the number of coalition structures grows as the number of agents increases and corresponds to $\sum_{i=1}^{n} Z(n,i)$, where $Z(n,i)$, also known as the Stirling number of the second kind, is the number of coalition structures with i coalitions, and may be computed using the following recurrence: $Z(n,i) = iZ(n-1,i) + Z(n-1,i-1)$, where $Z(n,n) = Z(n,1) = 1$. As proved in [2], the number of coalition structures is $O(n^n)$ and $\omega(n^{n/2})$, and hence an exhaustive enumeration becomes prohibitive.

In this paper we focus on games that are neither *superadditive* nor *subadditive* for which the problem of coalition structure generation is computationally complex. Indeed, for superadditive games where $v(S\cup T) \geq v(S)+v(T)$ (meaning any two disjoint coalitions are better off by merging together), and for subadditive games where $v(S\cup T) < v(S)+v(T)$ for all disjoint coalitions $S, T \subseteq A$, the problem of coalition structure generation is trivial. In particular, in superadditive games, the agents are better off forming the grand coalition where all agents operate together ($C^* = \{A\}$), while in subadditive games, the agents are better off by operating alone ($C^* = \{\{a_1\}, \{a_2\}, \ldots, \{a_n\}\}$).

3 Related Work

Previous works on CSG can be broadly divided into two main categories: exact algorithms that return an optimal solution, and approximate algorithms that find an approximate solution with limited resources.

A deterministic algorithm must systematically explore the search space of candidate solutions. One of the first algorithms returning an optimal solution is the dynamic programming algorithm (DP) proposed in [8] for the set partitioning problem.This algorithm is polynomial in the size of the input ($2^n - 1$) and it runs in $O(3^n)$ time, which is significantly less than an exhaustive enumeration ($O(n^n)$). However, DP is not an anytime algorithm, and has a large memory requirement. Indeed, for each coalition C it computes the tables $t_1(C)$ and $t_2(C)$. It computes all the possible splits of the coalition C and assigns to $t_1(C)$ the best split and to $t_2(C)$ its value. In [6] the authors proposed an improved version of the DP algorithm (IDP) performing fewer operations and requiring less memory than DP. IDP, as shown by the authors, is considered one of the fastest available exact algorithm in the literature computing an optimal solution.

Both DP and IDP are not anytime algorithms, they cannot be interrupted before their normal termination. In [2], Sandholm et al. have presented the first anytime algorithm, that can be interrupted to obtain a solution within a time limit but not guaranteed to be optimal. When not interrupted it returns the optimal solution. The CSG process can be viewed as a search in a coalition structure graph as reported in Figure 1. One desideratum is to be able to guarantee that the CS is within a worst case bound from optimal, i.e. that searching through a subset N of coalition structures, $k = \min\{k'\}$ where $k' \geq \frac{V(S^*)}{V(S^*_N)}$ is finite, and as small as possible, where S^* is the best CS and S^*_N is the best CS that has been seen in the subset N. In [2] has been proved that: a) to bound k, it suffices to search the lowest two levels of the coalition structure

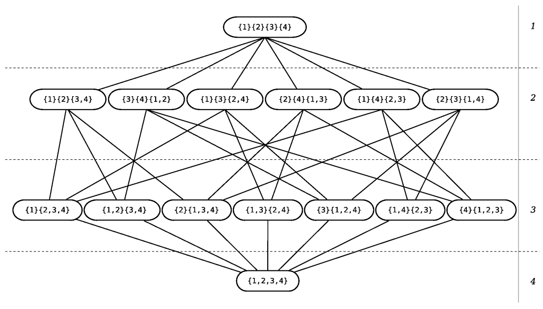

Fig. 1. Coalition structure graph for a 4-agent game.

graph (with this search, the bound $k = n$, and the number of nodes searched is 2^{n-1}); b) this bound is tight; and, c) no other search algorithm can establish any bound k while searching only 2^{n-1} nodes or fewer.

A new anytime algorithm has been proposed in [9], named IP, whose idea is to partition the space of the possible solutions into sub-spaces such that it is possible to compute upper and lower bounds on the values of the best CSs they contain. Then, these bounds are used to prune all the sub-spaces that cannot contain the optimal solution. Finally, the algorithm searches through the remaining sub-spaces adopting a branch-and-bound technique avoiding to examine all the solutions within the searched sub-spaces.

As regards the approximate algorithms, in [10] it has been proposed a solution based on a genetic algorithm, which performs well when there is some regularity in the search space. Indeed, the authors assume, in order to apply their algorithm, that the value of a coalition is dependent of other coalitions in the CS, making the algorithm not well suited for the general case. A new solution [11] is based on a Simulated Annealing algorithm [12], a widely used stochastic local search method. At each iteration the algorithm selects a random neighbour solution s' of a CS s. The search proceeds with an adjacent CS s' of the original CS s if s' yields a better social welfare than s. Otherwise, the search is continued with s' with probability $e^{(V(s')-V(s))/t}$, where t is the temperature parameter that decreases according to the annealing schedule $t = \alpha t$.

4 GRASP for CSG

The resource limits posed by MASs, such as the time for finding a solution, require to have approximation algorithms able to quickly find solutions that are within a specific factor of an optimal solution. In this section we present a new anytime algorithm for CSG based on a stochastic local search procedure, named GRASP-CSG.

A method to find high-quality solutions for a combinatorial problem is a two steps approach consisting of a greedy construction phase followed by a perturbative[1] local

[1] A perturbative local search changes candidate solutions by modifying one or more of the corresponding solution components.

Algorithm 1 GRASP-CSG

```
Require: $V$: the characteristic function; $A$: the set of $n$ agents; *maxiter*: maximum number of
  iterations
Ensure: solution $\widehat{C} \in \mathcal{M}(A)$
  $\widehat{C} = \emptyset, V(\widehat{C}) = -\infty$
  iter $= 0$
  while iter $<$ maxiter do
    $\alpha =$ rand(0,1);
    $C = \emptyset$; $i = 0$
    /* construction */
    while $i < n$ do
      $\mathcal{S} = \{C' | C' = add(C, A)\}$
      $\overline{s} = \max\{V(T) | T \in \mathcal{C}\}$
      $\underline{s} = \min\{V(T) | T \in \mathcal{C}\}$
      RCL $= \{C' \in \mathcal{S} | V(C') \geq \underline{s} + \alpha(\overline{s} - \underline{s})\}$
      randomly select an element $C$ from RCL
      $i \leftarrow i + 1$
    /* local search */
    $\mathcal{N} = \{C' \in neigh(C) | V(C') > V(C)\}$
    while $\mathcal{N} \neq \emptyset$ do
      select $C \in \mathcal{N}$
      $\mathcal{N} \leftarrow \{C' \in neigh(C) | V(C') > V(C)\}$
    if $V(C) > V(\widehat{C})$ then
      $\widehat{C} = C$
    iter = iter + 1
  return $\widehat{C}$
```

search [4]. Namely, the greedy construction method starts the process from an empty candidate solution and at each construction step adds the best ranked component according to a heuristic selection function. Successively, a perturbative local search algorithm is used to improve the candidate solution thus obtained. Advantages of this search method, over other stochastic local search algorithms, are the much better solution quality and fewer perturbative improvement steps to reach the local optimum. Greedy Randomized Adaptive Search Procedures (GRASP) [5] solve the problem of the limited number of different candidate solutions generated by a greedy construction search methods by randomising the construction method. GRASP is an iterative process, in which each iteration consists of a construction phase, producing a feasible solution, and a local search phase, finding a local optimum in the neighborhood of the constructed solution. The best overall solution is returned.

Algorithm 1 reports the outline for GRASP-CSG included in the ELK system. In each iteration, it computes a solution C by using a randomised constructive search procedure and then applies a local search procedure to C yielding an improved solution. The main procedure is made up of two components: a constructive phase and a local search phase. The constructive search algorithm used in GRASP-CSG iteratively adds a solution component by randomly selecting it, according to a uniform distribution, from a set, named *restricted candidate list* (RCL), of highly ranked solution components with respect to a greedy function $g : C \to \mathbb{R}$. The probabilistic component of GRASP-CSG is characterized by randomly choosing one of the best candidates in the RCL. In our case the greedy function g corresponds to the characteristic function V presented in Section 2. In particular, given V, the heuristic function, and $\mathcal{C}$, the set of feasible solution components, $\underline{s} = \min\{V(C)|C \in \mathcal{C}\}$ and $\overline{s} = \max\{V(C)|C \in \mathcal{C}\}$ are computed. Then the RCL is defined by including in it all the components C such that $V(C) \geq \underline{s}+\alpha(\overline{s}-\underline{s})$. The parameter α controls the amounts of greediness and randomness. A value $\alpha = 1$ corresponds to a greedy construction procedure, while $\alpha = 0$ produces a random construction. As reported in [13], GRASP with a fixed nonzero RCL parameter α is not asymptotically convergent to a global optimum. The solution to make the algorithm asymptotically globally convergent, could be to randomly select the parameter value from the continuous interval $[0, 1]$ at the beginning of each iteration and using this value during the entire iteration, as we implemented in GRASP-CSG.

Given a set of nonempty subsets of n agents A, $C = \{C_1, C_2, \dots, C_t\}$, such that $C_i \cap C_j \neq \emptyset$ and $\cup C \subset A$, the function $add(C, A)$ used in the construction phase returns a refinement C' obtained from C using one of the following operators:

1. $C' \to C \setminus \{C_i\} \cup \{C'_i\}$ where $C'_i = C_i \cup \{a_i\}$ and $a_i \notin \cup C$, or
2. $C' \to C \cup \{C_i\}$ where $C_i = \{a_i\}$ and $a_i \notin \cup C$.

Starting from the empty set, in the first iteration all the coalitions containing exactly one agent are considered and the best is selected for further specialization. At the iteration i, the working set of coalition C is refined by trying to add an agent to one of the coalitions in C or a new coalition containing the new agent is added to C.

To improve the solution generated by the construction phase, a local search is used. It works by iteratively replacing the current solution with a better solution taken from the neighborhood of the current solution while there is a better solution in the neighborhood. In order to build the neighborhood of a coalition structure C, *neigh(C)*, the following operators, useful to transform partitions of the grand coalition, have been used. Given a CS $C = \{C_1, C_2, \dots, C_t\}$:

split: $C \to C \setminus \{C_i\} \cup \{C_k, C_h\}$, where $C_k \cup C_h = C_i$, with $C_k, C_h \neq \emptyset$;
merge: $C \to C \setminus \{C_i, C_j\}_{i \neq j} \cup \{C_k\}$, where $C_k = C_i \cup C_j$;
shift: $C \to C \setminus \{C_i, C_j\}_{i \neq j} \cup \{C'_i, C'_j\}$, where $C'_i = C_i \setminus \{a_i\}$ and $C'_j = C_j \cup \{a_i\}$, with $a_i \in C_i$.
exchange: $C \to C \setminus \{C_i, C_j\}_{i \neq j} \cup \{C'_i, C'_j\}$, where $C'_i = C_i \setminus \{a_i\} \cup \{a_j\}$ and $C'_j = C_j \setminus \{a_j\} \cup \{a_i\}$, with $a_i \in C_i$ and $a_j \in C_j$;
extract: $C \to C \setminus \{C_i\}_{i \neq j} \cup \{C'_i, C_j\}$, where $C'_i = C_i \setminus \{a_i\}$ and $C_j = \{a_i\}$, with $a_i \in C_i$.

In the local search phase, the neighborhood of a coalition structure C is built by applying all the previous operators (split, merge, shift, exchange and extract) to C. As an example, in Table 1 is reported the application of the operators to the CS $\{\{12\}\{3\}\{4\}\}$. The problem in using more than the two classical merge and split operators corresponds to the fact of obtaining repetitions of the same CS. This problem deserves further attention, each operator must take into account how other operators works. Concerning the representation of the characteristic function and the search space, given n agents

Table 1. Operators applied to the CS $\{12\}\{3\}\{4\}$.

split	merge	shift	exchange	extract
$\{1\}\{2\}\{3\}\{4\}$	$\{123\}\{4\}$	$\{2\}\{13\}\{4\}$	$\{23\}\{1\}\{4\}$	$\{1\}\{2\}\{3\}\{4\}$
	$\{124\}\{3\}$	$\{2\}\{3\}\{14\}$	$\{24\}\{3\}\{1\}$	
	$\{12\}\{34\}$	$\{1\}\{23\}\{4\}$	$\{13\}\{2\}\{4\}$	
		$\{1\}\{3\}\{24\}$	$\{14\}\{3\}\{2\}$	

$A = \{a_1, a_2, \ldots, a_n\}$, we recall that the number of possible coalitions is $2^n - 1$. Hence, the characteristic function $v : 2^n \to \mathbb{R}$ is represented as a vector CF in the following way. Each subset $S \subseteq A$ (coalition) is described as a binary number $c_B = b_1 b_2 \cdots b_n$ where each $b_i = 1$ if $a_i \in S$, $b_i = 0$ otherwise. For instance, given $n = 4$, the coalition $\{a_2, a_3\}$ corresponds to the binary number 0110. Now, given the binary representation of a coalition S, its decimal value corresponds to the index in the vector CF where its corresponding value $v(S)$ is memorised. This gives us the possibility to have a random access to the values of the characteristic functions in order to efficiently compute the value V of a coalition structure.

Given a coalition structure $C = \{C_1, C_2, \ldots, C_k\}$, assuming that the C_i are ordered by their smallest elements, a convenient representation of the CS is a sequence $d_1 d_2 \cdots d_n$ where $d_i = j$, if the agent a_i belongs to the coalition C_j. Such sequences are known as *restricted growth sequences* [14] in the combinatorial literature. For instance, the sequence corresponding to the coalition structure $C = \{C_1, C_2, C_3\} = \{\{1, 2\}, \{3\}, \{4\}\}$ is 1123. Now in order to compute $V(C)$, we have to solve the sum $v(C_1) + v(C_2) + v(C_3)$, where C_1 corresponds to the binary number 1100, C_2 corresponds to the binary number 0010, and C_3 corresponds to the binary number 0001. Hence, $V(C) = v(C_1) + v(C_2) + v(C_3) = CF[12] + CF[2] + CF[1]$, where CF is the vector containing the values of the characteristic function.

5 Experimental results

In order to evaluate our GRASP-CSG, we implemented it in the C language and the corresponding source code has been included in the ELK system[2]. We also implemented the algorithm proposed by Sandholm et al. in [2], DP [8], and IDP [6], whose source code has been included in the ELK system. GRASP-CSG has been compared to those algorithms and to the Simulated Annealing algorithm (SA) proposed in [11], kindly provided by the authors.

In the following we present experimental results on the behaviour of these algorithms for some benchmarks considering solution qualities and the runtime performances. We firstly compared GRASP-CSG to DP and IDP. Then we evaluated its ability in generating solutions anytime when compared to the SA and to the Sandholm et al. algorithms.

Instances of the CSG problem have been defined using the following distributions for the values of the characteristic function:

[2] ELK is a system including many algorithms for the CSG problem whose source code is publicly available at http://www.di.uniba.it/∼ndm/elk/.

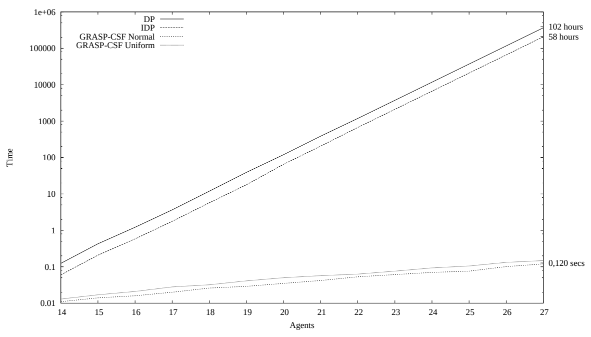

Fig. 2. Comparison between DP, IDP and GRASP-CSG.

1. Normal if $v(C) \sim N(\mu, \sigma^2)$ where $\mu = 1$ and $\sigma = 0.1$;
2. Uniform if $v(C) \sim U(a, b)$ where $a = 0$ and $b = 1$.

The algorithms are executed on PC with an Intel Core2 Duo CPU T7250 @ 2.00GHz and 2GB of RAM, running Linux kernel 2.6.31.

5.1 Optimal solution

Given different numbers of agents, ranging from 14 to 27, we compared GRASP-CSG to DP and IDP reporting the time required to find the optimal coalition structure. As reported in Figure 2, where the time in seconds is plotted in a log scale, GRASP-CSG outperforms both DP and IDP for all the distributions. Note that there is one line for DP and IDP since they do not depend on the input distribution but only on the input dimension. Over the problems, the execution time for DP (resp. IDP) ranges from 0.124 (resp. 0.06) seconds to approximately 366586 (resp. 207433) seconds, while for GRASP-CSG it ranges from 0.011 seconds to 0.12 seconds on average. In particular, for 27 agents GRASP-CSG is 1728608 times faster than IDP. As regards DP and IDP, due to high time consuming, values for problems with a number of agents ranging from 23 to 27 have been analytically calculated since their complexity linearly depends on the problem size. As we can see from Figure 2 this improvement grows as the dimension of the problem grows. Even if we cannot make a direct comparison to IP, as the authors reported in [9], IP is 570 times better than IDP in the case of uniform distribution for 27 agents. Since GRASP-CSG is 1728608 times faster than IDP for the same problem, we can argue that GRASP-CSG is faster than IP.

As regards GRASP-CSG we set the maxiter parameter to 20 even if in many cases the optimal CS has been found with fewer iterations, see the details in Figure 3. However, in this experiment this limit guarantees to find always the optimal coalition. Given the number of agents, 10 different instances of the problem for each distribution have been generated and time is averaged. Figure 3 reports an insight of the GRASP-CSG execution for three problems (18, 19 and 20 agents). For each iteration the graphs report the time and the relative quality of the solution averaged over 10 instances. The relative quality of a coalition structure C is obtained as $V(C)/V(C^*)$ where C^* is the optimal

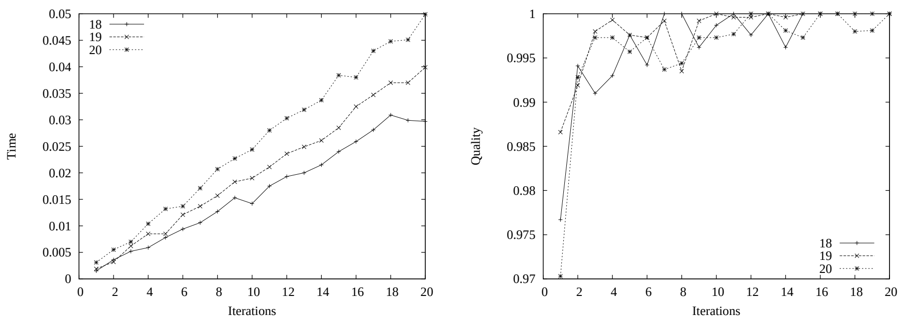

Fig. 3. Time (in seconds) and relative quality of GRASP-CSG obtained on the first 20 iterations.

CS. As we can see, the quality of the obtained CSs is very high just in the first iterations computed with few milliseconds.

5.2 Approximate solution

In this second experiment we compared the anytime characteristic of GRASP-CSG to the Sandholm et al. algorithm [2] and the Simulated Annealing algorithm [11]. We generated 10 instances for each problem with agents ranging from 14 to 20 and uniform distribution $U(0, 1)$. For each problem we set a time limit to return a good solution and we recorded the relative error of the obtained solution S by each of the three algorithms computed as $e = 1 - V(S)/V(S^*)$, where S^* is the best CS. Table 2 reports the error averaged over the 10 instances for each problem. As we can see GRASP-CSG is always able to find a better CS than those obtained by Sandholm et al. and SA. With this second experiment we can conclude that GRASP-CSG quickly finds very good solutions.

Table 2. Comparison of Sandholm et al., Simulated Annealing (SA) and GRASP-CSG.

Agents	Sandholm	SA	GRASP-CSG
14	0.1340	0.0320	0.0046
15	0.1862	0.0331	0.0000
16	0.1814	0.0205	0.0033
17	0.1514	0.0104	0.0000
18	0.1057	0.0001	0.0000
19	0.1393	0.0052	0.0005
20	0.1399	0.0021	0.0000

6 Conclusions

The paper presented the application of the stochastic local search GRASP to the problem of coalition structure generation. As reported in the experimental section the proposed algorithm outperforms some of the state of the art algorithms in computing optimal coalition structures.

As a future work it should be interesting to investigate the behaviour of the operators used to create the neighborhood of a coalition structure. In particular, an in deep study may be conducted in learning to choose the correct operators with respect to the distribution of the coalition values. Furthermore the application of shift, exchange and extract operators should generate repetitions of the same coalition structure obtained with the split and merge operators. Hence, an analysis on how to overcome this problem, avoiding to spend time and space resources, deserves more attention. Furthermore, we are planning to apply the proposed method to the more general games such as Partition Function Games [15], where the coalition's value may depend on the formation of another coalition, and to the task-based colaition formation problem [16].

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1990)
2. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F.: Coalition structure generation with worst case guarantees. Artificial Intelligence **111**(1-2) (1999) 209–238
3. Cramton, P., Shoham, Y., Steinberg, R., eds.: Combinatorial Auctions. MIT Press (2006)
4. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc. (2004)
5. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
6. Rahwan, T., Jennings, N.R.: An improved dynamic programming algorithm for coalition structure generation. In: Prooceedings of AAMAS08. (2008) 1417–1420
7. Kahan, J.P., Rapoport, A.: Theories of Coalition Formation. Lawrence Erlbaum Associates Publisher (1984)
8. Yeh, D.Y.: A dynamic programming approach to the complete set partitioning problem. BIT **26**(4) (1986) 467–474
9. Rahwan, T., Ramchurn, S.D., Jennings, N.R., Giovannucci, A.: An anytime algorithm for optimal coalition structure generation. Journal of Artificial Intelligence Research **34**(1) (2009) 521–567
10. Sen, S., Dutta, P.S.: Searching for optimal coalition structures. In: Prooceedings of ICMAS00, IEEE Computer Society (2000) 287–292
11. Keinänen, H.: Simulated annealing for multi-agent coalition formation. In: Prooceedings of KES-AMSTA09, Springer (2009) 30–39
12. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598) (1983) 671–680
13. Mockus, J., Eddy, E., Mockus, A., Mockus, L., Reklaitis, G.V.: Bayesian Heuristic Approach to Discrete and Global Optimization. Kluwer Academic Publishers (1997)
14. Milne, S.C.: Restricted growth functions, rank row matchings of partitions lattices, and q-stirling numbers. Advances in Mathemathics **43** (1982) 173–196
15. Rahwan, T., Michalak, T., Jennings, N., Wooldridge, M., McBurney, P.: Coalition structure generation in multi-agent systems with positive and negative externalities. In: Prooceedings of IJCAI-09. (2009) 257–263
16. Dang, V.D., Jennings, N.R.: Coalition structure generation in task-based settings. In: Proceeding of ECAI 2006, IOS Press (2006) 210–214