Artificial Intelligence 175 (2011) 2061–2074

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Randomized coalition structure generation

Travis Service [a,*], Julie Adams [b]

[a] Vanderbilt University, 357 Jacobs Hall, 2201 West End Nashville, TN 37235-1824, USA
[b] Vanderbilt University, 359 Jacobs Hall, 2201 West End Nashville, TN 37235-1824, USA

ARTICLE INFO

Article history:
Received 25 December 2010
Received in revised form 11 July 2011
Accepted 1 August 2011
Available online 9 August 2011

Keywords:
Coalition structure generation
Coalition formation
Characteristic function game

ABSTRACT

Randomization can be employed to achieve constant factor approximations to the coalition structure generation problem in less time than all previous approximation algorithms. In particular, this manuscript presents a new randomized algorithm that can generate a $\frac{2}{3}$ approximate solution in $O(\sqrt{n}2.587^n)$ time, improving upon the previous algorithm that required $O(\sqrt{n}2.83^n)$ time to guarantee the same performance. Also, the presented new techniques allow a $\frac{1}{4}$ approximate solution to be generated in the optimal time of $O(2^n)$ and improves on the previous best approximation ratio obtainable in $O(2^n)$ time of $\frac{1}{8}$. The presented algorithms are based upon a careful analysis of the sizes and numbers of coalitions in the smallest optimal coalition structures.

An empirical analysis of the new randomized algorithms compared to their deterministic counterparts is provided. We find that the presented randomized algorithms generate solutions with utility comparable to what is returned by their deterministic counterparts (in some cases producing better results on average). Moreover, a significant speedup was found for most approximation ratios for the randomized algorithms over the deterministic algorithms. In particular, the randomized $\frac{1}{2}$ approximate algorithm runs in approximately 22.4% of the time required for the deterministic $\frac{1}{2}$ approximation algorithm for problems with between 20 and 27 agents.

1. Introduction

Many situations require partitioning the agents into disjoint teams or coalitions where each coalition cooperatively completes a subgoal or subtask [1–3]. However, the process of optimally partitioning the agents into coalitions is computationally difficult due to the fact that the number of potential coalitions scales exponentially with the number of agents.

Characteristic function games, a class of cooperative games, models cooperative multi-agent situations by assigning each coalition a value indicating the joint utility those agents will receive if they form a coalition [1,2,4–7]. For example, this utility may be the difference between the utility earned by completing the subtask assigned to the potential coalition and the estimated cost incurred during task execution. Given a set of agents, N, a characteristic function game is defined by a function $v : N \rightarrow \mathbb{R}^{\geqslant 0}$. The value $v(C)$, for a coalition $C \subseteq N$, is the joint utility the members of C will receive if C forms. A coalition structure is simply a partitioning of the agents into disjoint coalitions. The *coalition structure generation problem* (CSG problem) is to construct a coalition structure CS that maximizes:

$$v(CS) = \sum_{C \in CS} v(C).$$

* Corresponding author.
E-mail addresses: tservice@acm.org (T. Service), julie.a.adams@vanderbilt.edu (J. Adams).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2011.08.002

The asymptotically fastest algorithm to solve the CSG problem exactly is based on dynamic programming and runs in $O(3^n)$ time [2,8]. Due to the high computational complexity of the optimal CSG algorithm, much research has focused on the development of anytime algorithms. While these anytime algorithms can quickly return approximate solutions, in the worst case most current anytime algorithms require $O(n^n)$ time to determine the optimal solution, far worse than the $O(3^n)$ time dynamic programming algorithm. Among the current state-of-the-art is Rahwan et al's. [1] Integer Partition algorithm.

Due to the high computational complexity of constructing the optimal coalition structure, our recent work has focused on developing approximation algorithms that return solutions with bounds on their quality in time less than $O(3^n)$ [9,10]. For example, our prior algorithm was capable of returning a solution that was at least $\frac{2}{3}$ of the optimal in $O(\sqrt{n}2.83^n)$ time.

This paper improves upon the state-of-the-art in CSG approximation algorithms by presenting randomized algorithms that achieve the same approximation ratios as our prior work, but require significantly less time. For example, our new randomized approximation algorithm is capable of generating a $\frac{2}{3}$ approximation in $O(\sqrt{n}2.59^n)$ time. This result improves upon our previous $O(\sqrt{n}2.83^n)$ time deterministic algorithm for the same approximation ratio. We also show how to generate a $\frac{1}{4}$ approximation in the optimal time of $O(2^n)$ (i.e., any algorithm must observe the values of all $2^n - 1$ coalitions in order to guarantee any approximate solution [2]).

Our methods also permit approximation ratios that are unachievable by current approximation algorithms. For example, our randomized algorithm is the first designed to guarantee a $\frac{3}{5}$ approximation ratio.

All of the presented approximation algorithms are probabilistic in the sense that they are guaranteed to return their stated approximation ratio with high probability. Iteratively running the algorithms a small number of times and simply taking the highest quality solution yields the stated approximation ratio with high probability. For example, if a given algorithm finds a $\frac{2}{3}$ approximation ratio with probability at least $\frac{1}{2}$, then running the algorithm k times and taking the best result yields a $\frac{2}{3}$ approximation with probability $1 - (\frac{1}{2})^k$. Several of the presented algorithmic results are of the form:

> Algorithm A returns a coalition structure that is guaranteed to have value within a factor of f of the optimal with probability greater than $\frac{1}{2}$. A runs in expected time $O(g(n))$.

That is, the output of the algorithm (and hence the quality of the solution it presents) is a random variable, and for some of the presented algorithms, the runtime of the algorithm is also a random variable.

An empirical study is presented that shows the randomized algorithms perform similarly to their deterministic counterparts in terms of utility and sometimes find higher quality solutions. Further, it is shown that the run time of the randomized algorithms, for most approximation guarantees, is significantly faster than the deterministic algorithms.

The remainder of the paper is organized as follows. Section 2 describes the prior work on coalition structure generation. Section 3 presents the new randomized algorithms for coalition structure generation. Section 4 provides an empirical comparison of the new randomized algorithms to their deterministic counterparts. Section 5 presents concluding remarks.

2. Related work

Much algorithm research has focused on the coalition structure generation problem. The current fastest algorithm that is guaranteed to find the optimal solution is based on dynamic programming and requires $O(3^n)$ time to find an optimal coalition structure on n agents [7,2,8]. Given this high computational complexity, recent work has focused on the development of anytime and approximation algorithms.

Sandholm et al. [2] developed one of the first anytime algorithms for the coalition structure generation problem. Sandholm et al. viewed coalition structure generation as a search through the lattice of partitions of the n agents, which they referred to as the coalition structure graph. Sandholm et al.'s algorithm proceeded by searching through the bottom two levels of the coalition structure graph (i.e., those coalition structures that consisted of only one or two coalitions), followed by a breadth first search from the top of the graph (i.e., examining those coalition structures that consist of i coalitions followed by those coalition structures that consist of $i - 1$ coalitions and so on). Sandholm et al. derive guarantees on the quality of the best solution found so far based upon which levels of the coalition structure graph have been searched.

Among the state-of-the-art in anytime coalition structure generation is Rahwan et al.'s [1,6] Integer Partition algorithm. As with Sandholm et al.'s algorithm, the Integer Partition algorithm searches directly through the space of coalition structures. The Integer Partition algorithm groups coalition structures together into subspaces based upon the sizes of the coalitions they contain. For example, both the coalition structures $\{\{1,2\},\{3\},\{4\}\}$ and $\{\{3,4\},\{1\},\{2\}\}$ are in the same subspace as they both contain a coalition of size 2 and two coalitions of size 1. Rahwan et al. show how to generate upper and lower bounds on the quality of the coalition structures in each subspace. The Integer Partition algorithm uses those bounds to perform a branch and bound search through the space of all coalition structures. Empirically, Rahwan et al. have shown that the Integer Partition algorithm is often capable of quickly pruning many of the subspaces from consideration and performs significantly better than Sandholm et al.'s algorithm.

While the Integer Partition algorithm performs well empirically on many problem distributions, it is possible to construct distributions on which the Integer Partition algorithms performance deteriorates significantly [10]. The worst case runtime required to find the optimal solution of both the Integer Partition algorithm and Sandholm et al.'s algorithm is $O(n^n)$, since in the worst case each coalition structure must be examined and there are $O(n^n)$ coalition structures.

2.1. Deterministic approximation algorithms

Our recent work has focused on developing approximation algorithms for coalition structure generation that are guaranteed to find an approximate solution of a given quality in less than $O(3^n)$ time [9,10]. For example, we have shown that it is possible to find an approximate solution with value at least $\frac{2}{3}$ of the optimal in $O(\sqrt{n}2.83^n)$ time. All current approximation algorithms are deterministic, in that they always return the same answer in the same amount of time when run on the same problem instance. The currently reported research expands upon the state-of-the-art by showing how to employ randomization to generate approximate solutions with the same quality guarantees in less time than the current deterministic techniques.

The presented randomized approximation algorithms all relying on a deterministic preprocessing phase. This preprocessing phase formed the basis for previous approximation algorithms.

The following definitions will be used:

Definition 2.1. A characteristic function v is *monotonic* iff whenever $C \subseteq S \subseteq N$, then $v(C) \leqslant v(S)$.

Intuitively, a characteristic function is monotonic if adding an agent to a coalition can never harm the coalition members.

Definition 2.2. A characteristic function v is *k-superadditive* iff for all $C, S \subseteq N$ such that

1. $C \cap S = \emptyset$ and
2. $|C \cup S| \leqslant k$,

then $v(C) + v(S) \leqslant v(C \cup S)$.

If v is n-superadditive (or just superadditive), then the merger of any two coalitions into a larger coalition is never harmful. If $CS^* = \{C_1, \ldots, C_k\}$ with $k \geqslant 2$ is an optimal coalition structure in a superadditive game, then $CS_2^* = \{C_1, \ldots, C_{k-1} \cup C_k\}$ will be optimal as well. This means that in a superadditive game, the coalition structure that places every agent into the same coalition (referred to as the grand coalition) is optimal.

Superadditivity is a natural assumption. If two coalitions C_1 and C_2 merge, in the worst case, the members of $C_1 \cup C_2$ behave as if the merger did not occur and receive value $v(C_1)+v(C_2)$. However, larger coalitions often times incur additional overhead (e.g., communication and/or computational overhead) [11]. It may be that the merger of coalitions is beneficial up to a point. That is, there is a point at which coordination costs outweigh the potential benefit derived from a larger coalition. The notion of k-superadditivity models such situations. For example, a game v may be k-superadditive, because even though $C_1 \cup C_2$ can earn strictly more utility than C_1 and C_2 independently, the cost of coordinating the actions of members of $C_1 \cup C_2$ outweighs the additional utility when $|C_1 \cup C_2| > k$. Our previous work showed that, as with superadditive games, the existence of optimal coalition structures with only a few coalitions in k-superadditive games can be guaranteed [9].

Theorem 2.3. *Let $r \leqslant n$ be a positive integer. If v is $\frac{n}{r}$-superadditive, then v has an optimal solution that consists of at most $2r-1$ coalitions. If v is $\frac{2n}{2r-1}$-superadditive, then v has an optimal solution consisting of $2r-2$ coalitions.*

The reader is directed to the prior paper [9] for the formal proof of Theorem 2.3; however, the intuition is as follows: Let v be k-superadditive and CS^* an optimal coalition structure in v that has the smallest number of coalitions. If CS^* contains two coalitions, C_1 and C_2, each of which contains no more than $\frac{k}{2}$ coalitions, then, as v is k-superadditive, $v(C_1 \cup C_2) \geqslant v(C_1) + v(C_2)$. However, a new optimal coalition structure with fewer coalitions can be constructed by replacing C_1 and C_2 in CS^* with $C_1 \cup C_2$, but this is a contradiction. Hence, at most one coalition in CS^* has no more than $\frac{k}{2}$ coalitions. Less formally, CS^* can have at most one *small* coalition (coalition with no more than $\frac{k}{2}$ agents). Therefore, most coalitions in CS^* must be *large* and there cannot be too many of them.

Our prior work showed how to construct a new game $v_{\frac{k}{r}}$ from any characteristic function game v such that:

1. $v_{\frac{k}{r}}$ is $\frac{kn}{r}$ superadditive and monotonic,
2. optimal solutions to v and $v_{\frac{k}{r}}$ have the same value, and
3. given an approximate solution to $v_{\frac{k}{r}}$, a solution of equal or greater value can be constructed in v in polynomial time.

Pseudo-code for the construction of a monotonic $v_{\frac{k}{r}}$ is broken up into two steps [9]. The first step, shown in Algorithm 2.1, takes an arbitrary coalitional game v and constructs a new game $v_{\frac{k}{r}}$ that is $\frac{kn}{r}$-superadditive, but not necessarily monotonic. The second step, shown in Algorithm 2.2, uses the newly constructed $v_{\frac{k}{r}}$ to construct a new game v_{max} that is both $\frac{kn}{r}$-superadditive and monotonic.

Algorithm 2.1 Constructing $v_{\frac{k}{r}}$.

```
for i = 1 to ⌊kn/r⌋ do
  for C ⊆ N, |C| = i do
    v_{k/r}(C) ← v(C)
    for C' ⊂ C do
      if v_{k/r}(C') + v_{k/r}(C \ C') > v_{k/r}(C) then
        v_{k/r}(C) ← v_{k/r}(C') + v_{k/r}(C \ C')
      end if
    end for
  end for
end for
v_{k/r}(C) = v(C) for all C such that |C| > kn/r
```

Algorithm 2.1 constructs a $\frac{kn}{r}$-superadditive game by considering the merger of every pair of disjoint coalitions, C and S, with $|C \cup S| \leqslant \frac{kn}{r}$ in order of increasing size. For each coalition $|C| \leqslant k$, the best value over all possible splittings of C into two disjoint coalitions is stored in $v_{\frac{k}{r}}(C)$. By proceeding in order of increasing size, this results in $v_{\frac{k}{r}}(C)$ equalling the value of the optimal coalition structure over the agents in C. For coalitions, C with more than k agents, $v_{\frac{k}{r}}(C)$ is set equal to $v(C)$. In addition to storing the value of each coalition, the optimal way in which to split a coalition C into two smaller coalitions $S, C \setminus S$ is stored as well. Given a coalition structure in $v_{\frac{k}{r}}$, a coalition of equal value can be constructed in v by following the splittings of coalitions.

Algorithm 2.2 Constructing v_{max}.

```
for k = 1 to n do
  for C ⊆ N, |C| = k do
    v_max(C) ← v_{k/r}(C)
    for a ∈ C do
      if v_max(C − {a}) > v_max(C) then
        v_max(C) ← v_max(C \ {a})
      end if
    end for
  end for
end for
```

Algorithm 2.2 constructs a monotonic game by setting for each $C \subseteq N$ $v_{max}(C) = \max_{S \subseteq C} v_{\frac{k}{r}}(S)$. Naively building v_{max} requires taking the maximum over $2^{|C|}$ values for each coalition C. Algorithm 2.2 constructs v_{max} more efficiently by considering coalitions in order of increasing size. When considering a coalition C, we have that $v_{max}(S) = \max_{S' \subseteq S} v_{\frac{k}{r}}(S')$ for all $S \subsetneq C$. Thus, determining $\max_{S \subseteq C} v_{\frac{k}{r}}(S)$ requires taking the maximum over $|C| + 1$ values $v_{\frac{k}{r}}(C)$ and $v_{\frac{k}{r}}(C \setminus a)$ for each $a \in |C|$. This total process requires $O(n2^n)$ time.

Notice that if it is known that v has an optimal solution, CS^* that consists of k solutions, then the coalition $C = \operatorname{argmax}_{C \subseteq N} v(C)$ has value at least $\frac{v(CS^*)}{k}$. This result is true since C has value at least as great as all coalitions in CS^*. Likewise if $C_1, \ldots, C_m$ are $m \leqslant k$ disjoint, possibly empty coalitions that maximize $v(C_1) + \cdots + v(C_m)$ over all disjoint sets of m, possible empty coalitions, then $v(C_1) + \cdots + v(C_m) \geqslant \frac{m}{k} v(CS^*)$. Given such a set of m coalitions, a coalition structure in the original game v that has value at least $\frac{m}{k} v(CS^*)$ can be constructed in polynomial time [9]. Algorithms that find such a collection of m coalitions will be referred to as solution extraction procedures.

Previous deterministic approximation algorithms worked by constructing $v_{\frac{k}{r}}$ from v and extracting two disjoint and possibly empty coalitions, C_1 and C_2, that maximized $v_{\frac{k}{r}}(C_1) + v_{\frac{k}{r}}(C_2)$. Since $v_{\frac{k}{r}}$ is monotonic, finding two such coalitions can be accomplished by taking the maximum of $v(C) + v(N \setminus C)$ over all $C \subseteq N$ in $O(2^n)$ time. This technique yields a $\frac{2}{2r-1}$ approximate solution when $v_{\frac{1}{r}}$ is constructed from v and a $\frac{1}{r-1}$ approximation solution when $v_{\frac{2}{2r-1}}$ is constructed.

3. Randomized coalition structure generation

This section presents a randomized algorithm for approximate coalition structure generation. The presented techniques significantly improve upon the runtimes of their deterministic counterparts.

The algorithms presented all succeed in finding an f-approximate solution with probability greater than $\frac{e-1}{e}$. Rerunning the algorithm, for example, three times and selecting the best solution found over all three runs, results in an f-approximate solution with probability at least $1 - (1 - \frac{e-1}{e})^3 > 0.95$ (i.e., the probability that all three runs fail to find an f-approximate solution is no more than $(1 - \frac{e-1}{e})^3$). However, the performance of the algorithm may not be the only probabilistic element. The algorithm's runtime may also be a random variable. When an algorithm's runtime is a random variable, we report the

algorithm's *expected* runtime. The observed runtime of the presented algorithms will be close to the expected runtime, with high probability.

All of the presented results follow from a careful analysis of the number of coalitions, and their respective sizes, for a particular optimal solution. As a result, a single randomized approximate algorithm that is capable of generating all the approximation ratios is not presented. Instead, it is necessary to tailor each algorithm to each approximation ratio. Therefore, a different algorithm for each approximation ratio is presented.

Many of the presented algorithms involve examining all coalitions of agents that contain at most some fixed fraction of the agents (i.e., all coalitions that consist of at most $\lfloor \frac{n}{r} \rfloor$ agents). The floor symbols are dropped from the statement of the algorithms and their correctness proofs in order to improve readability. For example, $\binom{n}{\frac{n}{r}}$ is written even though $\frac{n}{r}$ may not be an integer. The floor of $\frac{n}{r}$ is implicitly taken in these situations. Removal of the floor symbols affects only the readability of the formula and not the derived asymptotic bounds.

All the presented randomized approximation algorithms are based on the following underlying idea. Given an arbitrary coalitional game v, first construct $v_{\frac{k}{r}}$, for appropriately chosen k and r. Second, extract m disjoint coalitions that maximize their combined value. It has been previously shown how this can be accomplished in $O(n2^n)$ time for $m = 1, 2$. The main contribution of this work is to show that in some situations, extracting m disjoint, possible empty coalitions that maximize the sum of their values for $m = 3, 4$ is possible using simple randomized algorithms.

In summary, the presented randomized approximation techniques are variations of the following procedure:

1. Construct $v_{\frac{k}{r}}$ to ensure that there exists an optimal coalition structure that contains only m coalitions (for some constant m).
2. Run, possibly multiple solution extraction procedures and return the best solution.

Multiple solution extraction procedures are required because the smallest optimal solution may contain $1, \ldots, m$ coalitions. Each extraction procedure is designed to handle some subset of the possible sizes of the smallest optimal solution.

Constructing $v_{\frac{k}{r}}$ requires $O(\frac{\sqrt{n}r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}}(r-k)^{\frac{(r-k)n}{r}}})$ time [9]. All but one of the presented algorithms will require $v_{\frac{k}{r}}$ to be monotonic. Enforcing the constraint that $v_{\frac{k}{r}}$ is monotonic requires an additional $O(n2^n)$ time [9]. Unless otherwise stated, we assume that $v_{\frac{k}{r}}$ is monotonic.

Algorithm 3.1 represents the first solution extraction procedure to extract two disjoint coalitions C_1 and C_2 that maximize $v_{\frac{k}{r}}(C_1) + v_{\frac{k}{r}}(C_2)$.

Algorithm 3.1 Extracting two coalitions from a monotonic v.

```
1: C_1 = N
2: C_2 = ∅
3: for C ⊆ N do
4:     if v_max(C) + v_max(N \ C) > v_max(C_1) + v_max(C_2) then
5:         C_1 = C
6:         C_2 = N \ C
7:     end if
8: end for
```

The second and third extraction procedures (Algorithms 3.2 and 3.3) are randomized and require a priori knowledge of the maximum sizes of the coalitions to be extracted.

Algorithm 3.2 works as follows: Assume that it is known there are three disjoint coalitions C_1^*, C_2^* and C_3^* with total value at least f times that of an optimal solution. A simple randomized approach to find three disjoint coalitions with total value at least f times that of the optimal simply adds each agent to one of three sets with equal probability. That is, with probability $\frac{1}{3}$ each agent a_i is added to coalition C_1, with probability $\frac{1}{3}$ a_i is added to C_2, and with probability $\frac{1}{3}$ to C_3. Since v_{max} is monotonic, as long as $C_i^* \subseteq C_i$ for $i = 1, 2, 3$, then the total value of C_1, C_2 and C_3 will be at least f times that of an optimal solution. The probability that $C_i^* \subseteq C_i$ for $i = 1, 2$, and 3 is simply $(\frac{1}{3})^{|C_1^*|+|C_2^*|+|C_3^*|}$. Lemma 3.1 shows that in order to be guaranteed to find C_1, C_2 and C_3 with high probability, it is necessary to repeat this procedure $O(3^{|C_1^*|+|C_2^*|+|C_3^*|})$ times.

Lemma 3.1. *In a binomial distribution with success probability p after $\lceil \frac{1}{p} \rceil$ trials, the probability of at least one success is at least $\frac{e-1}{e}$ (approximately,* 63.21%*).*

Please see Appendix A for the proof of Lemma 3.1.

Lemma 3.2 provides a proof of the correctness and runtime of Algorithm 3.2.

Algorithm 3.2 Extracting three coalitions from a monotonic ν. Requires one parameter: *num_iterations*.

$C_1 \leftarrow \emptyset$
$C_2 \leftarrow \emptyset$
$C_3 \leftarrow \emptyset$
for $i = 0$ to *num_iterations* **do**
  $C'_1 \leftarrow \emptyset$
  $C'_2 \leftarrow \emptyset$
  $C'_3 \leftarrow \emptyset$
  for $k = 0$ to *num_agents* **do**
    Add agent a_k to one of the coalitions C'_1, C'_2 or C'_3 with equal probability.
  end for
  if $\nu(C'_1) + \nu(C'_2) + \nu(C'_3) > \nu(C_1) + \nu(C_2) + \nu(C_3)$ **then**
    $C_1 \leftarrow C'_1$
    $C_2 \leftarrow C'_2$
    $C_3 \leftarrow C'_3$
  end if
end for

Algorithm 3.3 Extracting four coalitions from a monotonic ν. Requires two parameters: *num_iterations* and *upperbound*.

$C_i \leftarrow \emptyset$ for $i = 1, 2, 3, 4$
for $i = 0$ to *num_iterations* **do**
  $C'_i \leftarrow \emptyset$ for $i = 1, 2, 3, 4$
  for $k = 0$ to *num_agents* **do**
    Add agent a_k to one of S_1 and S_2 with equal probability
  end for
  for $A \subset S_1$, $|A| \leqslant$ *upperbound* **do**
    if $\nu(A) + \nu(S_1 \setminus A) > \nu(C_1) + \nu(C_3)$ **then**
      $C'_1 \leftarrow S_1 \setminus A$
      $C'_3 \leftarrow A$
    end if
  end for
  for $A \subset S_2$, $|A| \leqslant$ *upperbound* **do**
    if $\nu(A) + \nu(S_2 \setminus A) > \nu(C_2) + \nu(C_4)$ **then**
      $C'_2 \leftarrow S_2 \setminus A$
      $C'_4 \leftarrow A$
    end if
  end for
  if $\nu(C'_1) + \nu(C'_2) + \nu(C'_3) + \nu(C'_4) > \nu(C_1) + \nu(C_2) + \nu(C_3) + \nu(C_4)$ **then**
    $C_i \leftarrow C'_i$ for $i = 1, 2, 3, 4$
  end if
end for

Lemma 3.2. *Let C_1^*, C_2^* and C_3^* be three disjoint coalitions from a set of n agents. If ν is monotonic, then after $O(3^{|C_1^*|+|C_2^*|+|C_3^*|})$ iterations the best solution found by Algorithm* 3.2 *has value at least $\nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*)$ with probability at least $\frac{e-1}{e}$.*

Proof. During a single iteration of Algorithm 3.2, the probability that all agents in C_i^* are placed in C'_i, for $i = 1, 2$, and 3, is simply $(\frac{1}{3})^{|C_1^*|+|C_2^*|+|C_3^*|}$. The iterations of Algorithm 3.2 form a binomial distribution with success probability $(\frac{1}{3})^{|C_1^*|+|C_2^*|+|C_3^*|}$. Therefore, after $3^{|C_1^*|+|C_2^*|+|C_3^*|}$ iterations, with probability greater than $\frac{e-1}{e}$, Algorithm 3.2 succeeds in finding three coalitions $C'_1 \subseteq C_1^*$, $C'_2 \subseteq C_2^*$ and $C'_3 \subseteq C_3^*$. Since ν is monotonic, $\nu(C'_1) + \nu(C'_2) + \nu(C'_3) \geqslant \nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*)$. □

The following variation of Algorithm 3.2 (henceforth referred to as Algorithm 3.2(a)) was found to be more efficient in practice. Rather then place each agent in C'_1, C'_2 or C'_3 with equal probability, each agent a_i is placed in C'_1 with probability 0.375 and in C'_2 and C'_3 with probability 0.3125. The increased efficiency stems from the fact that the probabilities 0.375 and 0.3125 can be created easily from a fair coin, while generating an event with $\frac{1}{3}$ probability cannot. All of our empirical results employ Algorithm 3.2(a). The use of Algorithm 3.2(a) does not affect the presented asymptotic results. Lemma 3.3 shows that the required number of iterations increase only slightly under in Algorithm 3.2(a).

Lemma 3.3. *Let C_1^*, C_2^* and C_3^* be three disjoint coalitions from a set of n agents. If ν is monotonic, then after $O(3.012^{|C_1^*|+|C_2^*|+|C_3^*|})$ iterations the best solution found by Algorithm* 3.2(a) *has value at least $\nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*)$ with probability at least $\frac{e-1}{e}$.*

Proof. During a single iteration of Algorithm 3.2(a), the probability that all agents in C_i^* are placed in C'_i, for $i = 1, 2$, and 3, is simply $0.375^{|C_1^*|} \cdot 0.3125^{|C_2^*|+|C_3^*|}$. The iterations of Algorithm 3.2(a) form a binomial distribution with success probability $0.375^{|C_1^*|} \cdot 0.3125^{|C_2^*|+|C_3^*|}$. Therefore, after $(\frac{1}{0.375})^{|C_1^*|} \cdot (\frac{1}{0.3125})^{|C_2^*|+|C_3^*|}$ iterations, with probability greater than $\frac{e-1}{e}$, Algorithm 3.2(a) succeeds in finding three coalitions $C'_1 \subseteq C_1^*$, $C'_2 \subseteq C_2^*$ and $C'_3 \subseteq C_3^*$. Since ν is monotonic: $\nu(C'_1) + \nu(C'_2) + \nu(C'_3) \geqslant \nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*)$.

Assume without loss of generality that $|C_1| \geqslant |C_2| \geqslant |C_3|$. Thus, $|C_2| + |C_3| \leqslant (\frac{2}{3})^{|C_1|+|C_2|+|C_3|}$. Therefore, as $\frac{1}{0.375} < \frac{1}{0.3125}$ the total number of iterations required by the variation on Algorithm 3.2(a) is at most:

$$\left(\frac{1}{0.375}\right)^{\frac{|C_1|+|C_2|+|C_3|}{3}} \cdot \left(\frac{1}{0.3125}\right)^{\frac{2(|C_1|+|C_2|+|C_3|)}{3}} = O\left(3.012^{(|C_1|+|C_2|+|C_3|)}\right). \qquad \square$$

The fourth and final solution extraction procedure (Algorithm 3.3) extracts four disjoint coalitions from a monotonic game. Algorithm 3.3 works as follows: Assume that it is known that there are four disjoint coalitions C_1^*, C_2^*, C_3^* and C_4^* with total value at least f times that of the optimal. Without loss of generality, assume that $|C_1^*| \geqslant |C_2^*| \geqslant |C_3^*| \geqslant |C_4^*|$. Algorithm 3.3 places each agent randomly in one of two sets, S_1 or S_2, with equal probability. After randomly distributing the agents among S_1 and S_2, Algorithm 3.3 finds the two coalitions $C_{S_1} \subset S_1$ that maximizes $v(C_{S_1}) + v(S_1 \setminus C_{S_1})$ subject to $|C_{S_1}| \leqslant u$. Notice that, since v is monotonic, this is equivalent to finding the two coalitions $C_1, C_2 \subset S_1$ that maximize $v(C_1) + v(C_2)$ subject to at least one of the coalitions containing no more than u agents. This same process is repeated for S_2. That is, the coalition $C_{S_2} \subset S_2$ that maximizes $v(C_{S_2}) + v(S_2 \setminus C_{S_2})$ subject to $|C_{S_2}| \leqslant u$ is found. Algorithm 3.3 returns the maximum over all iterations of $v(C_{S_1}) + v(S_1 \setminus C_{S_1}) + v(C_{S_2}) + v(S_2 \setminus C_{S_2})$.

If $C_1^*, C_3^* \subset S_1$, $C_2^*, C_4^* \subset S_2$ and $u \geqslant |C_3^*| \geqslant |C_4^*|$, then $v(C_{S_1}) + v(S_1 \setminus C_{S_1}) \geqslant v(C_1^*) + v(C_3^*)$ and $v(C_{S_2}) + v(S_2 \setminus C_{S_2}) \geqslant v(C_2^*) + v(C_4^*)$. Therefore, Algorithm 3.3 will return an f approximate solution.

Determining the runtime of Algorithm 3.3 requires the following combinatorial identity, the proof of which is provided in Appendix A.

Lemma 3.4. *The following identity holds for all positive integers $n > u$:*

$$\sum_{k=u}^{n} \binom{n}{k}\binom{k}{r} = 2^{n-u}\binom{n}{u}.$$

Lemma 3.5 bounds the time required per iteration for Algorithm 3.3.

Lemma 3.5. *With upperbound $u \leqslant \frac{n}{3}$, each iteration of Algorithm* 3.3 *runs in time $O(\frac{1}{2^u}\binom{n}{u})$.*

Proof. The probability that $|S_1| = k$ in any given iteration is $\binom{n}{k} \cdot \frac{1}{2^n}$, as there are $\binom{n}{k}$ possible sets of k agents and the probability of S_1 being any given subset of the agents is $\frac{1}{2^n}$.

If $|S_1| = k$, then the for loop beginning on line 7 is executed 2^k times if $k \leqslant u$ and $\binom{k}{u}$ times if $k > u$. Likewise, the for loop beginning on line 13 runs for 2^{n-k} iterations if $n - k \leqslant u$ and $\binom{n-k}{u}$ iterations if $n - k > u$.

Summing over all possible sizes of S_1 yields an expected runtime of:

$$\begin{aligned}
&\frac{1}{2^n}\sum_{k=0}^{u}\binom{n}{k}\left[2^k + \binom{n-k}{u}\right] + \frac{1}{2^n}\sum_{k=u+1}^{n-u-1}\binom{n}{k}\left[\binom{k}{u} + \binom{n-k}{u}\right] + \frac{1}{2^n}\sum_{k=n-u}^{n}\binom{n}{k}\left[\binom{k}{u} + 2^{n-k}\right] \\
&= \frac{1}{2^n}\sum_{k=0}^{n-u-1}\binom{n}{k}\binom{n-k}{u} + \frac{1}{2^n}\sum_{k=u+1}^{n}\binom{n}{k}\binom{k}{u} + \frac{1}{2^n}\sum_{k=0}^{u}\binom{n}{k}2^k + \frac{1}{2^n}\sum_{k=n-u}^{n}\binom{n}{k}2^{n-k} \\
&= \frac{2}{2^n}\sum_{k=u}^{n}\binom{n}{k}\binom{k}{u} + \frac{2}{2^n}\sum_{k=0}^{u}\binom{n}{k}2^k \leqslant \frac{2}{2^n}2^{n-u}\binom{n}{u} + n\frac{2}{2^n}2^u\binom{n}{u} = O\left(\frac{1}{2^u}\binom{n}{u}\right). \qquad \square
\end{aligned}$$

Lemma 3.6. *Let C_1^*, C_2^*, C_3^* and C_4^* be four disjoint coalitions from a set of n agents such that $|C_1^*| \geqslant |C_2^*| \geqslant |C_3^*| \geqslant |C_4^*|$. If v is monotonic, then after $O(2^{|C_1^*|+|C_2^*|+|C_3^*|+|C_4^*|})$ iterations the best solution found by Algorithm* 3.3 *with an upperbound of at least $|C_3^*|$ has value at least $v(C_1^*) + v(C_2^*) + v(C_3^*) + v(C_4^*)$ with probability at least $\frac{e-1}{e}$.*

Proof. If during some iteration of Algorithm 3.3, $C_1^*, C_3^* \subseteq S_1$ and $C_2^*, C_4^* \subseteq S_2$, then since *upperbound* $\geqslant |C_3^*| \geqslant |C_4^*|$ in the for loop beginning on line 7, eventually $A = C_3^*$. Since v is monotonic and $C_1^* \subset S_1 \setminus C_3^*$, then $v(S_1 \setminus C_3^*) + v(C_3^*) \geqslant v(C_1^*) + v(C_3^*)$ and $v(C_1') + v(C_3') \geqslant v(C_1^*) + v(C_3^*)$.

Likewise, since *upperbound* $\geqslant |C_4^*|$, then $v(C_2') + v(C_4') \geqslant v(C_2^*) + v(C_4^*)$. Therefore, Algorithm 3.3 will return a solution with value at least $v(C_1^*) + v(C_2^*) + v(C_3^*) + v(C_4^*)$.

By Lemma 3.1, the probability that during at least one iteration of Algorithm 3.3 $C_1, C_3 \subseteq S_1$ and $C_2, C_4 \subseteq S_2$ is strictly greater than $\frac{e-1}{e}$. $\square$

Table 1
Approximation ratios and corresponding running times for the presented algorithm for arbitrary coalitional games. The first column provides the approximate ratio of the algorithm in terms of the fraction of the value of the optimal solution guaranteed. The second and third columns list the time required to obtain the stated approximation ratio for the deterministic and randomized algorithms respectively.

Approximation ratio	Deterministic alg. runtime	Randomized alg. runtime
2/3	$O(\sqrt{n}2.83^n)$	$O(\sqrt{n}2.587^n)$
3/5	N/A	$O(\sqrt{n}2.382^n)$
1/2	$O(\sqrt{n}2.59^n)$	$O(\sqrt{n}2.09^n)$
1/3	$O(\sqrt{n}2.22^n)$	$O(n2^n)$
1/4	$O(n2^n)$	$O(2^n)$

Lemma 3.7 is used to determine the runtime of the various approximation algorithms. The proof of Lemma 3.7 can be found in Appendix A.

Lemma 3.7. *Let r be a fixed integer, then* $\binom{n}{n/r} = O\left(\frac{r^n}{\sqrt{n}(r-1)^{\frac{(r-1)n}{r}}}\right)$.

The new randomized approximation algorithms can be presented given the provided lemmas. Table 1 presents a summary of the approximate ratios and running times obtainable by the new randomized algorithms.

Theorem 3.8. *A $\frac{2}{3}$ approximate coalition structure in v can be found in $O(\sqrt{n}2.587^n)$ time.*

Proof. A $\frac{2}{3}$ approximation can be obtained as follows: First construct $v_{\frac{2}{5}}$ in $O(\sqrt{n}2.587^n)$ time, then run Algorithm 3.1 to find two disjoint coalitions that maximize the sum of their values. Finally run Algorithm 3.2 for $3^{\frac{4n}{5}} = O(2.41^n)$ iterations. This procedure guarantees a $\frac{2}{3}$ approximate solution with high probability.

Let CS^* be an optimal coalition structure in $v_{\frac{2}{5}}$ with the least number of coalitions. By Theorem 2.3, $|CS^*| \leqslant 4$. We consider two cases.

1. $|CS^*| \leqslant 3$. Algorithm 3.1 returns a $\frac{2}{3}$ approximate solution in $O(2^n)$ time.
2. $|CS^*| = 4$. Let the coalitions in CS^* be $|C_1| \geqslant |C_2| \geqslant |C_3| \geqslant |C_4|$. Since $v_{\frac{2}{5}}$ is $\frac{2n}{5}$ superadditive, $|C_3| + |C_4| > \frac{2n}{5}$. Otherwise $\{C_1, C_2, C_3 \cup C_4\}$ is an optimal coalition structure with fewer coalitions. Since $|C_3| \geqslant |C_4|$, it must be the case that $|C_3| > \frac{n}{5}$. Therefore, $|C_1|, |C_2| > \frac{n}{5}$.
   At least one of the following must be true:
   (a) $v(C_1) + v(C_2) \geqslant \frac{2}{3}v(CS^*)$,
   (b) $v(C_1) + v(C_3) + v(C_4) \geqslant \frac{2}{3}v(CS^*)$ or
   (c) $v(C_2) + v(C_3) + v(C_4) \geqslant \frac{2}{3}v(CS^*)$.
   Since, if all three inequalities were false, summing the right- and left-hand sides yields:

$$2\big(v(C_1) + v(C_2) + v(C_3) + v(C_4)\big) < 2 * v\big(CS^*\big).$$

   However, this is a contradiction as $v(C_1) + v(C_2) + v(C_3) + v(C_4) = v(CS^*)$.

If $v(C_1) + v(C_2) \geqslant \frac{2}{3}v(CS^*)$, then Algorithm 3.1 returns a $\frac{2}{3}$ approximate solution in $O(2^n)$ time. If one of the second or third cases is true, then both $|C_1| + |C_3| + |C_4|$ and $|C_2| + |C_3| + |C_4|$ contain fewer than $\frac{4n}{5}$ coalitions. Therefore, by Lemma 3.2, Algorithm 3.2 returns a $\frac{2}{3}$ approximate solution in $O(3^{\frac{4n}{5}}) = O(2.41^n)$ iterations.

The bottleneck step is constructing $v_{\frac{2}{5}}$, which requires $O(\sqrt{n}2.587^n)$ time. □

Theorem 3.9. *A $\frac{3}{5}$ approximate coalition structure in v can be found in $O(\sqrt{n}2.38^n)$ time.*

Proof. First construct $v_{\frac{1}{3}}$ in $O(\sqrt{n}2.38^n)$ time. After constructing $v_{\frac{1}{3}}$, run Algorithm 3.1 followed by Algorithm 3.2 for $3^{\frac{3n}{4}} = O(2.28^n)$ iterations.

Let CS^* be an optimal coalition structure in $v_{\frac{1}{3}}$ with the least number of coalitions. By Theorem 2.3, $|CS^*| \leqslant 5$. There are three cases to consider.

1. $|CS^*| \leqslant 3$. Algorithm 3.1 will return a $\frac{2}{3} > \frac{3}{5}$ approximate solution in $O(2^n)$ time.

2. $|CS^*| = 4$. If $v(C_2) + v(C_3) + v(C_4) < \frac{3}{5}v(CS^*)$, then $v(C_1) > \frac{2}{5}v(CS^*)$. Let C_i be the coalition in $\{C_2, C_3, C_4\}$ with the highest value. Since $v(C_2) + v(C_3) + v(C_4) = v(CS^*) - v(C_1)$, then $v(C_i) \geqslant \frac{1}{3}(v(CS^*) - v(C_1))$. Therefore,

$$v(C_i) + v(C_1) \geqslant \frac{v(CS^*)}{3} + \frac{2}{3}v(C_1) \geqslant \frac{v(CS^*)}{3} + \frac{4}{15}v\big(CS^*\big) = \frac{9}{15}v\big(CS^*\big) = \frac{3}{5}v\big(CS^*\big).$$

   Hence, either $v(C_2) + v(C_3) + v(C_4) \geqslant \frac{3}{5}v(CS^*)$ or $v(C_1) + v(C_i) \geqslant \frac{3}{5}v(CS^*)$. If $v(C_2) + v(C_3) + v(C_4) \geqslant \frac{3}{5}v(CS^*)$, then Algorithm 3.2 returns a $\frac{3}{5}$ approximate solution in $O(3^{\frac{3n}{4}}) = O(2.28^n)$ time because $|C_1| \geqslant \frac{n}{4}$. If $v(C_1) + v(C_i) \geqslant \frac{3}{5}v(CS^*)$ then Algorithm 3.1 returns a $\frac{3}{5}$ approximate solution.
3. $|CS^*| = 5$. Since v is $\frac{n}{3}$ superadditive, any three coalitions C_i, C_j and C_k can contain at most $\frac{2n}{3}$ agents. Since the three highest valued coalitions in CS^* must be a $\frac{3}{5}$ approximate solution, then Algorithm 3.2 returns a $\frac{3}{5}$ approximate solution in $O(3^{\frac{2n}{3}}) = O(2.09^n)$ iterations. □

Theorem 3.10. *A $\frac{1}{2}$ approximate coalition structure in v can be found in $O(\sqrt{n}2.09^n)$ expected time.*

Proof. Construct $v_{\frac{1}{4}}$ in $O(\sqrt{n}2.09^n)$ time, then run Algorithm 3.1 to identify two disjoint coalitions that maximize the sum of their values, followed by Algorithm 3.2 for $3^{\frac{5n}{8}} = O(1.99^n)$ iterations. Finally, run Algorithm 3.3 for $2^{\frac{4n}{7}} = O(1.49^n)$ iterations with *upperbound* $= \frac{n}{6}$. By Lemma 3.5, each iteration of Algorithm 3.3 requires $O\big(\frac{1}{2^{\frac{n}{6}}}\binom{n}{n/6}\big)$ time. Hence, the total runtime of Algorithm 3.3 is $O\big(\frac{2^{\frac{4n}{7}}}{2^{\frac{n}{6}}}\binom{n}{n/6}\big) = O(2.08^n)$.

We show that this procedure guarantees a $\frac{1}{2}$ approximate solution with high probability.

Let CS^* be an optimal coalition structure in $v_{\frac{1}{4}}$ with the least number of coalitions. By Theorem 2.3, $|CS^*| \leqslant 7$.

Let $|C_1| \geqslant |C_2| \geqslant \cdots \geqslant |C_r|$ be the coalitions in CS^*. Note that $|C_{r-1}| + |C_r| > \frac{n}{4}$ otherwise, since $v_{\frac{1}{4}}$ is $\frac{n}{4}$ superadditive, $\{C_1, \ldots, C_{r-2}, C_{r-1} \cup C_r\}$ is an optimal coalition structure with fewer coalitions. Since $|C_{r-1}| \geqslant |C_r|$, it must be that $|C_{r-1}| \geqslant \frac{n}{8}$. Hence, for all $i < r$, $|C_i| \geqslant \frac{n}{8}$.

There are four cases to be considered.

1. $|CS^*| \leqslant 4$. Algorithm 3.1 returns a $\frac{1}{2}$ approximate solution in $O(2^n)$ time.
2. $|CS^*| = 5$. One of $\{C_1, C_2\}$ and $\{C_3, C_4, C_5\}$ must have total value at least $\frac{v(CS^*)}{2}$. If $v(C_1) + v(C_2) \geqslant \frac{v(CS^*)}{2}$, then Algorithm 3.1 returns a $\frac{1}{2}$ approximation solution. Note that $|C_1| + |C_2| \geqslant \frac{2n}{5}$ (as they are the two largest of the 5 coalitions in CS^*) and $|C_3| + |C_4| + |C_5| \leqslant \frac{3n}{5}$. Therefore, if $v(C_3) + v(C_4) + v(C_5) \geqslant \frac{v(CS^*)}{2}$, then Algorithm 3.2 returns a solution with value at least $v(C_3) + v(C_4) + v(C_5) \geqslant \frac{v(CS^*)}{2}$ with high probability in $O(3^{\frac{3n}{5}}) = O(1.94^n)$ iterations.
3. $|CS^*| = 6$. One of $\{C_1, C_5, C_6\}$ and $\{C_2, C_3, C_4\}$ must have total value at least $\frac{v(CS^*)}{2}$. If $\{C_1, C_5, C_6\}$ has total value at least $\frac{v(CS^*)}{2}$, then $|C_1| + |C_5| + |C_6| \leqslant \frac{5}{8}$ because $|C_2|, |C_3|, |C_4| \geqslant \frac{n}{8}$. Likewise, if $\{C_2, C_3, C_4\}$ has total value at least $\frac{v(CS^*)}{2}$, then $|C_2| + |C_3| + |C_4| \leqslant \frac{5}{8}$ because $|C_5| + |C_6| \geqslant \frac{n}{4}$ and $|C_1| \geqslant \frac{n}{8}$. Therefore, in either case, Algorithm 3.2 returns a solution with value at least $\frac{v(CS^*)}{2}$.
4. $|CS^*| = 7$. One of $\{C_1, C_2, C_3\}$ and $\{C_4, C_5, C_6, C_7\}$ must have total value at least $\frac{v(CS^*)}{2}$. Since $|C_4| + |C_5| + |C_6| + |C_7| \geqslant \frac{1}{2}$ then $|C_1| + |C_2| + |C_3| \leqslant \frac{1}{2}$. Therefore, if $\{C_1, C_2, C_3\}$ has value at least $\frac{v(CS^*)}{2}$, then Algorithm 3.2 will return a $\frac{1}{2}$ approximate solution in $O(3^{\frac{n}{2}}) = O(1.74^n)$ iterations. Otherwise, if $\{C_4, C_5, C_6, C_7\}$ has value at least $\frac{v(CS^*)}{2}$, then Algorithm 3.3 will return a $\frac{1}{2}$ approximate solution in $2^{\frac{4n}{7}}$ iterations because $|C_7| \leqslant |C_6| \leqslant \frac{1}{6}$. □

Theorem 3.11. *A $\frac{1}{3}$ approximate coalition structure in v can be found in $O(n2^n)$ time.*

Proof. First construct $v_{\frac{2}{9}}$ in $O(n2^n)$ time. Run Algorithm 3.1 to identify two disjoint coalitions that maximize the sum of their values, followed by Algorithm 3.2 for $3^{\frac{5n}{12}} = O(1.59^n)$ iterations.

We show that this procedure guarantees a $\frac{1}{3}$ approximate solution with high probability.

Let CS^* be an optimal coalition structure in $v_{\frac{2}{9}}$ with the least number of coalitions. By Theorem 2.3, $|CS^*| \leqslant 8$. There are three cases to consider.

1. $|CS^*| \leqslant 6$. Algorithm 3.1 returns a $\frac{1}{3}$ approximate solution in $O(2^n)$ time.
2. $|CS^*| = 7$. One of $\{C_1, C_2\}$, $\{C_3, C_4\}$, and $\{C_5, C_6, C_7\}$ must have total value at least $\frac{v(CS^*)}{3}$. If $\{C_1, C_2\}$ or $\{C_3, C_4\}$ has total value at least $\frac{v(CS^*)}{3}$, then Algorithm 3.1 returns a $\frac{1}{3}$ approximation solution. Otherwise, Algorithm 3.2 returns a $\frac{1}{3}$ approximation solution in $O(3^{\frac{3n}{7}}) = O(1.61^n)$ iterations.

3. $|CS^*| = 8$. One of $\{C_1, C_2\}$, $\{C_3, C_4, C_5\}$, and $\{C_6, C_7, C_8\}$ must have total value at least $\frac{v(CS^*)}{3}$. If $\{C_1, C_2\}$ has value at least $\frac{v(CS^*)}{3}$, then Algorithm 3.1 will return a $\frac{1}{3}$ approximate solution. Notice that $|C_1| + |C_2| \geqslant \frac{2n}{8} = \frac{n}{4}$ and $v_{\frac{2}{9}}$ is $\frac{2n}{9}$ superadditive, $|C_7| + |C_8| > \frac{2n}{9}$. Therefore, $|C_6| + |C_7| + |C_8| \geqslant \frac{n}{3}$. Thus, $n - \frac{n}{4} - \frac{n}{3} = \frac{5n}{12} \geqslant |C_3| + |C_4| + |C_5| \geqslant |C_6 + |C_7| + |C_8|$. Therefore, if either $\{C_3, C_4, C_5\}$ or $\{C_6, C_7, C_8\}$ has total value at least $\frac{v(CS^*)}{3}$, then Algorithm 3.2 will find a $\frac{1}{3}$ approximation in $O(3^{\frac{5n}{12}}) = O(1.59^n)$ iterations. □

The presented $\frac{1}{4}$ approximation algorithm runs in $O(2^n)$ time. Since requiring $v_{\frac{k}{r}}$ to be monotonic requires $O(n2^n)$ time, it is not possible to use monotonicity to simplify the $\frac{1}{4}$ approximation algorithm. Algorithm 3.4 presents a randomized algorithm to extract two disjoint coalitions that maximize the sum of their values in games that are not necessarily monotonic and forms the basis of the $\frac{1}{4}$ approximation algorithm.

Algorithm 3.4 works similarly to Algorithm 3.3. Each agent is placed into one of two sets S_1 or S_2 with equal probability. For each of S_1 and S_2 the highest valued coalition containing at most *upperbound* agents is determined. Each iteration of this process yields two disjoint coalitions C_1' and C_2'. The two coalitions with the highest total value are returned. The analysis of the running time and correctness of Algorithm 3.4 is analogous to that of Algorithm 3.3. Hence, each iteration of Algorithm 3.4 requires $O\left(\frac{1}{2^n}\binom{n}{u}\right)$ time for an upperbound of u. Therefore if C_1 and C_2 are two disjoint coalitions each containing fewer than *upperbound* agents, then after $2^{|C_1|+|C_2|}$ iterations Algorithm 3.4 finds an approximate solution with value at least $v(C_1) + v(C_2)$ with probability $\frac{e-1}{e}$.

Algorithm 3.4 Extracting two coalitions from a not necessarily monotonic v. Requires two parameters: *num_iterations* and *upperbound*.

```
 1: C_i ← ∅ for i = 1, 2
 2: for i = 0 to num_iterations do
 3:     C'_i ← ∅ for i = 1, 2
 4:     for k = 0 to num_agents do
 5:         Add agent a_k to one of S_1 and S_2 with equal probability
 6:     end for
 7:     for C ⊂ S_1, |C| ⩽ upperbound do
 8:         if v(C) > v(C'_1) then
 9:             C'_1 ← C
10:         end if
11:     end for
12:     for C ⊂ S_2, |C| ⩽ upperbound do
13:         if v(C) > v(C'_2) then
14:             C'_2 ← C
15:         end if
16:     end for
17:     if v(C'_1) + v(C'_2) > v(C_1) + v(C_2) then
18:         C_i ← C'_i for i = 1, 2
19:     end if
20: end for
```

Theorem 3.12. *A $\frac{1}{4}$ approximate coalition structure in v can be found in $O(2^n)$ expected time.*

Proof. First, construct $v_{\frac{2}{9}}$ in $O(2^n)$ time. Note that in this situation $v_{\frac{2}{9}}$ is not necessarily monotonic. Once $v_{\frac{2}{9}}$ is constructed, identify the coalition $C \subseteq N$ that has the highest value (this can be done in a single $O(2^n)$ time loop through all coalitions). Finally, run Algorithm 3.4 twice. First for $2^{\frac{4n}{9}} = O(1.36^n)$ iterations with an upperbound of $\frac{2n}{9}$, for a total running time of $O(1.96^n)$. Second for $2^{\frac{2n}{5}} = O(1.32^n)$ iterations with an upperbound of $\frac{n}{4}$, for a total running time of $O(1.95^n)$.

Let CS^* be an optimal coalition structure in $v_{\frac{1}{4}}$ with the fewest number of coalitions. By Theorem 2.3, $|CS^*| \leqslant 8$. Assume, without loss of generality that for $C_i, C_j \in CS^*$, $|C_i| \geqslant |C_j|$ whenever $i < j$. There are five cases to consider.

1. $|CS^*| \leqslant 4$. The highest valued coalition is a $\frac{1}{4}$ approximate solution and is found in $O(2^n)$ time.
2. $|CS^*| = 5$. One of C_1, C_2, C_3 and $\{C_4, C_5\}$ must have total value at least $\frac{v(CS^*)}{4}$. Since $|C_4| + |C_5| \leqslant \frac{2n}{5}$ and $|C_5| \leqslant |C_4| \leqslant \frac{n}{4}$ after $2^{\frac{2n}{5}}$ iterations, Algorithm 3.4 will find a $\frac{1}{4}$ approximate solution.
3. $|CS^*| = 6$. One of C_1, C_2, $\{C_3, C_4\}$ and $\{C_5, C_6\}$ must have total value at least $\frac{v(CS^*)}{4}$. Since any two coalitions C_i and C_j must collectively contain at least $\frac{2n}{9}$ agents and $|C_1| + |C_2| \geqslant \frac{n}{3}$, both $|C_3| + |C_6|$ and $|C_4| + |C_5|$ are bounded above by $n - \frac{2n}{9} - \frac{n}{3} = \frac{4n}{9}$. Note that,

$$3 \cdot |C_3| \leqslant |C_1| + |C_2| + |C_3| = n - |C_4| - |C_5| - |C_6| \leqslant n - \frac{n}{3} = \frac{2n}{3},$$

Table 2
The average utilities generated by the randomized and deterministic algorithms for five different approximation ratios on three problem distributions.

	Approximation ratio	2/3	3/5	1/2	1/3	1/4
Normal	Randomized	32.497	32.401	32.466	32.371	28.568
	Deterministic	32.575	N/A	32.497	32.375	32.371
Modified	Randomized	617.14	610.1	529.71	432.585	316.59
Uniform	Deterministic	619.51	N/A	503.595	405.79	385.415
NDCS	Randomized	62.246	61.5042	57.466	49.087	36.772
	Deterministic	63.163	N/A	52.648	48.788	48.072

where the last inequality follows from the fact that $|C_5| + |C_6| \geqslant \frac{2n}{9}$. Therefore, $|C_3| \leqslant \frac{2n}{9}$. Hence, Algorithm 3.4 will find a $\frac{1}{4}$ approximate solution after $2^{\frac{4n}{9}}$ iterations.

4. $|CS^*| = 7$. One of C_1, $\{C_2, C_3\}$, $\{C_4, C_5\}$ and $\{C_6, C_7\}$ must have total value at least $\frac{v(CS^*)}{4}$. Again, since any two coalitions C_i and C_j must collectively contain at least $\frac{2n}{9}$ agents and $|C_1| \geqslant \frac{n}{7}$, then $|C_2| + |C_7|$, $|C_3| + |C_6|$ and $|C_4| + |C_5|$ are bounded above by $n - \frac{4n}{9} - \frac{n}{7} = \frac{26n}{63}$. Notice that,

$$2 \cdot |C_2| \leqslant |C_1| + |C_2| = n - |C_3| - |C_4| - |C_5| - |C_6| - |C_7| \leqslant n - \frac{5n}{9} = \frac{4n}{9}.$$

As before, the last inequality follows from the fact that $v_{\frac{2}{9}}$ is $\frac{2n}{9}$-superadditive. Therefore, $|C_2| \leqslant \frac{2n}{9}$ and Algorithm 3.4 will find a $\frac{1}{4}$ approximate solution after $2^{\frac{26n}{63}}$ iterations.

5. $|CS^*| = 8$. One of $\{C_1, C_8\}$, $\{C_2, C_7\}$, $\{C_3, C_6\}$ and $\{C_4, C_5\}$ must have total value at least $\frac{v(CS^*)}{4}$. Again, since any two coalitions C_i and C_j must collectively contain at least $\frac{2n}{9}$ agents, then $|C_1| + |C_8|$, $|C_2| + |C_7|$, $|C_3| + |C_6|$ and $|C_4| + |C_5|$ are bounded above by $n - \frac{6n}{9} = \frac{n}{3}$. The coalitions other than C_1 must contain at least $\frac{7n}{9}$ agents, since $v_{\frac{2}{9}}$ is $\frac{2n}{9}$-superadditive. Therefore, C_1 can contain at most $\frac{2n}{9}$ agents. Therefore, Algorithm 3.4 will find a $\frac{1}{4}$ approximate solution after $2^{\frac{n}{3}}$ iterations. □

4. Empirical study

This section presents an empirical comparison of the randomized approximation algorithms to their deterministic counterparts. The empirical results are presented in two subsections. The first compares the randomized and deterministic algorithms in terms of the solution utilities they generate. The second subsection compares the runtimes of the randomized and deterministic algorithms over a number of problem sizes. All data points are averaged over 20 independent runs. For each run, the randomized algorithms were run for three iterations to guarantee that the stated approximation ratio was found with probability $1 - (1 - \frac{e-1}{e})^3 > 0.95$.

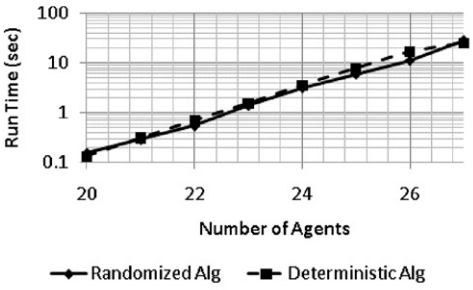
4.1. Utility comparison

All experiments presented in this subsection consists of problems with 24 agents. The performance of the randomized and deterministic algorithms are compared over the following three problem distributions, which are standard to the coalition structure generation literature [1,9,10,12]:

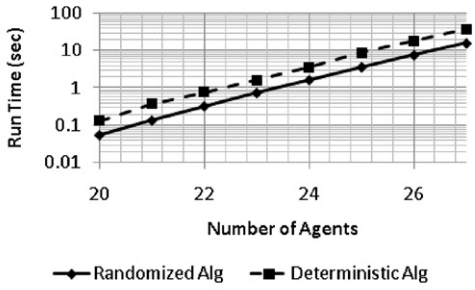
1. Normal Distribution: the value of each coalition C was set to $|C| \cdot \mathcal{N}(1.0, 0.01)$.
2. Modified Uniform Distribution: each coalition C is assigned a value drawn uniformly between 0 and $10 \cdot |C|$; however, each coalition's value is increased by a random number drawn uniformly between 0 and 50 with 20% probability.
3. Normally Distributed Coalition Structures (NDCS): the value of each coalition C is drawn from a normal distribution with mean $|C|$ and standard deviation $\sqrt{|C|}$.

Table 2 provides the average utility of the solutions generated by both the deterministic and the randomized algorithms. Recall that no deterministic $\frac{3}{5}$ approximation algorithm exists. In general, the average utility of solutions generated by the randomized algorithm was slightly less than that of the deterministic algorithm. This is a result of the randomized algorithm being probabilistic. When averaged over 20 independent runs, it is likely during at least one run that the algorithm will fail to find a solution as good as the deterministic algorithm's solution.

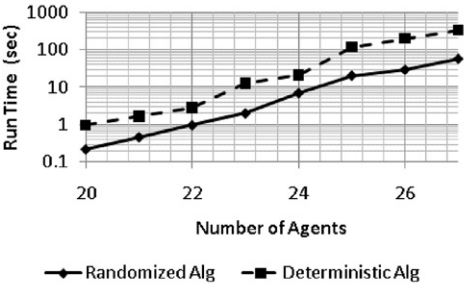
It is interesting to note that during some of the runs the randomized algorithm performed better than the deterministic algorithm. For example, the average performance of the randomized $\frac{1}{3}$ approximation algorithm was slightly higher then the average performance of the deterministic $\frac{1}{3}$ approximation for the Modified Uniform problem distribution. This is to be expected though as the following example illustrates. Consider the $\frac{2}{3}$ approximation algorithms. Assume that the optimal coalition structure consists of three equally sized, and equally valued, coalitions, C_1,C_2 and C_3. Further assume that all



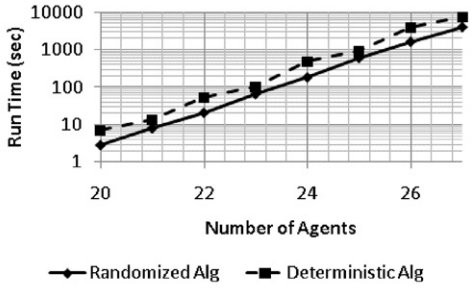
(a) Runtime in seconds of the both the deterministic and randomized $\frac{1}{4}$ approximation algorithms.



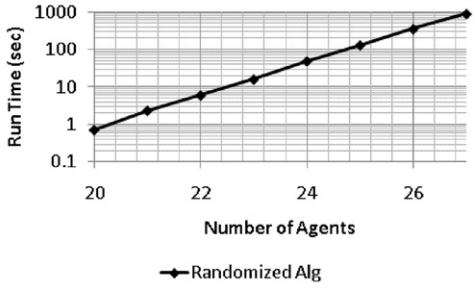
(b) Runtime in seconds of the both the deterministic and randomized $\frac{1}{3}$ approximation algorithms.



(c) Runtime in seconds of the both the deterministic and randomized $\frac{1}{2}$ approximation algorithms.



(d) Runtime in seconds of the both the deterministic and randomized $\frac{2}{3}$ approximation algorithms.



(e) Runtime in seconds of the randomized $\frac{3}{5}$ approximation algorithm.

Fig. 1. Comparison of the runtimes of the deterministic and randomized algorithms for each approximation guarantee on a variety of problem sizes. Runtimes are plotted on a logarithmic scale.

other coalitions have value 0. The deterministic $\frac{2}{3}$ approximation algorithm will simply return an approximate coalition structure consisting of two of these coalitions, say C_1 and C_2. However, it is possible that the randomized $\frac{2}{3}$ approximation algorithm will return a coalition structure consisting of C_1, C_2 and C_3, thus returning a solution with value strictly greater than that returned by the deterministic algorithm. While this example is artificial, it illustrates one type of situation where the randomized algorithms can outperform their deterministic counterparts (i.e., situations where the randomized algorithm has some probability of returning an optimal solution while the deterministic algorithm is unable too).

4.2. Runtime comparison

This subsection presents a comparison of the runtimes of the deterministic and the randomized algorithms. Fig. 1 presents a comparison of the runtimes of both the randomized and deterministic algorithms for all performance guarantees on problems ranging from 20 to 27 agents. As the run time of both the randomized and deterministic algorithms does not depend on the problem instance (i.e., the particular coalition utilities) the reported running times are for only the modified uniform distribution.

As can be seen in Fig. 1, in most cases the runtime of the randomized algorithms was lower than that of their deterministic counterparts. The exception is the $\frac{1}{4}$ approximation algorithms in which the randomized and deterministic algorithms performed roughly equivalently in terms of runtime (however, the randomized algorithm was slightly faster on average for most problem sizes).

The following average speedups were achieved by the randomized algorithms compared to their deterministic counterparts.

1. The $\frac{1}{4}$ randomized algorithm ran in 89.6% of the time as its deterministic counterpart.
2. The $\frac{1}{3}$ randomized algorithm ran in 42.7% of the time as its deterministic counterpart.
3. The $\frac{1}{2}$ randomized algorithm ran in 22.4% of the time as its deterministic counterpart.
4. The $\frac{2}{3}$ randomized algorithm ran in 50.8% of the time as its deterministic counterpart.

The empirically determined runtime of the randomized algorithms scaled more quickly than the theoretical runtime analysis provided in Section 3. This is believed to be a result of terms hidden in the $O(\cdot)$ notation. With larger numbers of agents, the empirical runtime should match the theoretical bounds.

5. Conclusions

This paper presents simple randomized algorithms for generating coalition structures, along with quality guarantees, for arbitrary and monotonic coalitional games. The presented algorithms generate the same approximation ratios in less time than all previous approximation algorithms. The algorithms build on prior work and follow from a careful analysis of the number of coalitions, and their sizes, in particular coalition structures. The probability that any given randomized algorithm generates a solution within the stated approximate ratio can be made arbitrarily close to 1 by rerunning the algorithm a small number of times. For example, 3 runs guarantee that the stated approximation ratio is achieved with probability greater than 95%.

One of the presented randomized algorithms can obtain a $\frac{2}{3}$ approximate in $O(\sqrt{n}2.587^n)$ time, improving upon the previous approximation algorithm that required $O(2.83^n)$ time to generate a $\frac{2}{3}$ approximate solution. Also the new techniques allow a $\frac{1}{4}$ approximate solution to be generated in the optimal time of $O(2^n)$ (i.e., in order to make any guarantees on solution quality the values of all $2^n - 1$ coalitions must be observed at least once). This result improves on the previous best approximation ratio obtainable in $O(2^n)$ time of $\frac{1}{8}$.

This manuscript also shows how to employ randomization to generate performance guarantees that current deterministic approaches are unable to generate. The presented randomized algorithms are the first algorithms capable of guaranteeing an approximation ratio of $\frac{3}{5}$ in arbitrary characteristic function games.

A limitation of the techniques presented in this manuscript is that the desired approximation ratio must be chosen a priori and then the appropriate algorithm run. Thus, the present work stands in contrast to previous coalition structure generation work where a single anytime algorithm is presented and generates successively better solution quality guarantees. As this paper has demonstrated, the anytime property can be sacrificed for better solution quality guarantees in less time when the necessary approximation ratio is known ahead of time. However, the requirement of selecting an approximation ratio a priori is common to approximation algorithms for many combinatorial problems.

Appendix A. Proofs

Lemma 3.1. *In a binomial distribution with success probability p, after $\lceil \frac{1}{p} \rceil$ trials the probability of at least one success is at least $\frac{e-1}{e}$ (approximately,* 63.21%*).*

Proof. Since the probability that any single trial is unsuccessful is $1 - p$, the probability that n trials are all unsuccessful is $(1 - p)^n$. Hence, the probability of at least one successful trial in n trials is $1 - (1 - p)^n$.

Consider the case were $n = \lceil \frac{1}{p} \rceil$, then

$$(1 - p)^n = \left(1 - \frac{1}{n}\right)^n.$$

Since $(1 - \frac{1}{n})^n$ increases monotonically to $\frac{1}{e}$ as $n \to \infty$, $(1 - \frac{1}{n})^n \leqslant \frac{1}{e}$. Hence, $1 - (1 - p)^{\frac{1}{p}} \geqslant \frac{e-1}{e}$. □

Lemma 3.4. *The following identity holds for all positive integers $n > u$:*

$$\sum_{k=u}^{n} \binom{n}{k}\binom{k}{u} = 2^{n-u}\binom{n}{u}.$$

Proof. Let $f(x) = (x+1)^n$. $f(x) = \sum_{k=0}^{n} \binom{n}{k} x^k$, by the binomial theorem. Differentiating $f(x)$ u times yields:

$$\begin{aligned} f^{(u)}(x) &= n \cdot (n-1) \cdot \cdots \cdot (n-u+1)(x+1)^{n-u+1} \\ &= \sum_{k=u}^{n} \binom{n}{k} k \cdot (k-1) \cdot \cdots \cdot (k-u+1) x^{k-u+1}, \end{aligned}$$

which can be rewritten as:

$$\begin{aligned} f^{(u)}(x) &= \frac{n!}{(n-u)!}(x+1)^{n-u} \\ &= \sum_{k=u}^{n} \binom{n}{k} \frac{k!}{(k-u)!} x^{k-u}. \end{aligned}$$

Finally, dividing both sides by $u!$ and setting $x = 1$ yields the desired identity. □

Lemma 3.7. *Let r be a fixed integer, then* $\binom{n}{n/r} = O\left(\frac{r^n}{\sqrt{n}(r-1)^{\frac{(r-1)n}{r}}}\right)$.

Proof. Recall that Stirling's approximation states:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \big(1 + o(1)\big) = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right),$$

$$\begin{aligned} \binom{n}{n/r} &= \frac{n!}{\frac{n}{r}!\frac{(r-1)n}{r}!} \\ &= O(1) \cdot \left(\frac{\sqrt{n}(\frac{n}{e})^n}{\sqrt{\frac{n}{r}}(\frac{\frac{n}{r}}{e})^{\frac{n}{r}} \cdot \sqrt{n - \frac{n}{r}}(\frac{(n-\frac{n}{r})}{e})^{(n-\frac{n}{r})}}\right) \\ &= O(1) \cdot \left(\frac{n^n}{\sqrt{n}(\frac{n}{r})^{\frac{n}{r}} \cdot (n - \frac{n}{r})^{(n-\frac{n}{r})}}\right) \\ &= O\left(\frac{r^n}{\sqrt{n}(r-1)^{\frac{(r-1)n}{r}}}\right). \qquad \square \end{aligned}$$

References

[1] T. Rahwan, S. Ramchurn, N. Jennings, A. Giovannucci, An anytime algorithm for optimal coalition structure generation, Journal of Artificial Intelligence Research 34 (2009) 521–567.

[2] T. Sandholm, K. Larson, M. Anderson, O. Shehory, F. Tohmé, Coalition structure generation with worst case guarantees, Artificial Intelligence 111 (1–2) (1999) 209–238. URL http://jmvidal.cse.sc.edu/library/sandholm99b.pdf.

[3] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) (1998) 165–200. URL http://jmvidal.cse.sc.edu/library/shehory98a.pdf.

[4] V.D. Dang, N. Jennings, Generating coalition structures with finite bound from the optimal guarantees, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004, pp. 564–571.

[5] T. Rahwan, N.R. Jennings, An improved dynamic programming algorithm for coalition structure generation, in: Proceedings of the 7th International Conference on, Autonomous Agents and Multi-Agent Systems, 2008, pp. 1417–1420. URL http://eprints.ecs.soton.ac.uk/15062/.

[6] T. Rahwan, N. Jennings, Coalition structure generation: Dynamic programming meets anytime optimisation, in: Proceedings of the 23rd Conference on Artificial Intelligence (AAAI), 2008, pp. 156–161. URL http://eprints.ecs.soton.ac.uk/16112/.

[7] T. Rahwan, N. Jennings, An improved dynamic programming algorithm for coalition structure generation, in: Proceedings of the 7th International Conference on, Autonomous Agents and Multi-Agent Systems, 2008, pp. 1417–1420. URL http://eprints.ecs.soton.ac.uk/15062/.

[8] Y. Yeh, A dynamic programming approach to the complete set partitioning problem, BIT Numerical Mathematics 26 (4) (1986) 467–474.

[9] T. Service, J. Adams, Approximate coalition structure generation, in: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI), 2010.

[10] T. Service, J. Adams, Constant factor approximation algorithms for coalition structure generation, Autonomous Agents and Multi-Agent Systems, 2010. URL http://dx.doi.org/10.1007/s10458-010-9124-7.

[11] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, M. Yokoo, Coalition structure generation utilizing compact characteristic function representations, in: Proceedings of the 15th international Conference on Principles and Practice of Constraint Programming, CP'09, 2009, pp. 623–638.

[12] K. Larson, T. Sandholm, Anytime coalition structure generation: An average case study, Journal of Experimental and Theoretical AI 12 (2000) 40–47.