Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Approximate Coalition Structure Generation

Travis C. Service and **Julie A. Adams**
Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, Tennessee, USA
{travis.c.service,julie.a.adams}@vanderbilt.edu

Abstract

Coalition formation is a fundamental problem in multi-agent systems. In characteristic function games (CFGs), each coalition C of agents is assigned a value indicating the joint utility those agents will receive if C is formed. CFGs are an important class of cooperative games; however, determining the optimal coalition structure, partitioning of the agents into a set of coalitions that maximizes the social welfare, currently requires $O(3^n)$ time for n agents.

In light of the high computational complexity of the coalition structure generation problem, a natural approach is to relax the optimality requirement and attempt to find an approximate solution that is guaranteed to be close to optimal. Unfortunately, it has been shown that guaranteeing a solution within any factor of the optimal requires $\Omega(2^n)$ time. Thus, the best that can be hoped for is to find an algorithm that returns solutions that are guaranteed to be as close to the optimal as possible, in as close to $O(2^n)$ time as possible.

This paper contributes to the state-of-the-art by presenting an algorithm that achieves better quality guarantees with lower worst case running times than all currently existing algorithms. For example, our algorithm improves the previous best approximation ratio of $\frac{1}{2}$ obtainable in $O(\sqrt{n}2.83^n)$ time to $\frac{2}{3}$ and obtains a $\frac{1}{2}$ approximation in $O(\sqrt{n}2.59^n)$. Our approach is also the first algorithm to guarantee a constant factor approximation ratio, $\frac{1}{8}$, in the optimal time of $O(2^n)$. The previous best ratio obtainable in $O(2^n)$ was $\frac{2}{n}$.

Introduction

Coalition formation is a central issue in systems with cooperating agents (Sandholm et al. 1999; Rahwan et al. 2009). Coalition formation is often studied in characteristic function games (CFG), a class of cooperative games, where each coalition is assigned a value indicating the joint utility those agents will receive if they form a coalition. Given a set of agents N, a CFG is defined on N by a function $\nu : N \rightarrow \mathcal{R}^{\geq 0}$. For a coalition $C \subseteq N$, $\nu(C)$ is interpreted as the utility the members of C will receive if C forms. The coalition structure generation problem is then to find a partitioning of N, or a coalition structure, CS that maximizes:

$$\sum_{C \in CS} \nu(C).$$

Much prior work has investigated algorithmic aspects of the coalition structure generation problem (Sandholm et al. 1999; Rahwan et al. 2009; Rahwan and Jennings 2008a; 2008b; Larson and Sandholm 2000; Dang and Jennings 2004). Roughly speaking, most recent coalition structure generation algorithmic work can be partitioned into two categories: design-to-time algorithms and anytime algorithms.

Design-to-time algorithms are guaranteed to find the optimal solution, but must be run to completion to do so. The best design-to-time algorithms, which are based on dynamic programming, have the current fastest worst case run time ($O(3^n)$ for n agents) of all coalition structure generation algorithms. However, for large collections of agents there may not be sufficient time to run the design-to-time algorithms to completion (Sandholm et al. 1999; Yeh 1986; Rahwan and Jennings 2008b).

Anytime algorithms are capable of returning approximation solutions, along with guarantees on the quality of the solution, without running to completion. Anytime algorithms can quickly return approximate solutions; however, in the worst case most current anytime algorithms require $O(n^n)$ time to find the optimal solution, far worse than the best design-to-time algorithms. Among the current state-of-the-art are Rahwan et al's. (Rahwan et al. 2009) Integer Partition algorithm and our previous anytime dynamic programming algorithm (Service and Adams 2010).

In light of the high computational complexity of constructing the optimal coalition structure, a natural approach is to relax the optimality requirement and attempt to identify an approximate solution that is guaranteed to be within some constant factor of the optimal. Unfortunately, Sandholm et al. (Sandholm et al. 1999) show that in order to make any guarantee on the quality of the solution returned, in the worst case, any algorithm must examine the value of each of the $O(2^n)$ possible coalitions. Thus, the runtime of any coalition structure generation algorithm that provides guarantees on the quality of the solutions it produces must be $\Omega(2^n)$. The best that can be hoped for is to find an algorithm that returns solutions that are guaranteed to be as close to the optimal as possible (i.e., factors close to 1) in as close to $O(2^n)$ time as possible.

Recently we developed an algorithm that is able to guarantee constant factor approximation ratios in less than $O(3^n)$ time (Service and Adams 2010). However, our previ-

ous algorithm is only capable of guaranteeing solution qualities of up to $\frac{1}{2}$ of the optimal.

This paper contributes to the state-of-the-art by presenting an algorithm that achieves better quality guarantees with lower worst case running times than all currently existing algorithms. In particular, our technique is capable of generating a solution with $\frac{2}{3}$ of the optimal in $O(\sqrt{n}2.83^n)$ time and a solution with value within $\frac{1}{2}$ of the optimal in $O(\sqrt{n}2.59^n)$ time, improving upon the ratio's obtainable in our previous work. Our new algorithm is also the first to be capable of guaranteeing a constant factor approximation ratio in the optimal time of $O(2^n)$. Our algorithm is able to guarantee a solution with a value within $\frac{1}{8}$ of the optimal in $O(2^n)$ time. This result greatly improves the previous best guarantee obtainable in general characteristic function games in $O(2^n)$ of $\frac{2}{n}$ (Sandholm et al. 1999).

Our approach to approximate coalition structure generation is similar to our previous work, but is based on an entirely different principle. Like our previous algorithm, our new technique works by extracting approximate solutions from partially completed dynamic programming tables. However, our new approach is capable of generating better quality guarantees than our previous algorithm in the same amount of time.

Approximation Technique

The presented approach to obtaining constant factor approximations of the optimal coalition structure in an arbitrary characteristic function ν requires applying two transformations to ν, one after the other, to obtain new characteristic functions ν' and ν'' such that:

1. the values of the optimal solutions in ν, ν' and ν'' are the same,
2. given a coalition structure, not necessarily optimal, in ν' or ν'', a coalition structure of equal, or greater, value can be constructed in ν in polynomial time, and
3. ν' and ν'' both have optimal solutions consisting of a constant number, m, of coalitions.

Since ν' has an optimal solution consisting of m coalitions, the highest valued coalition, C, in ν' will have value at least $\frac{1}{m}$ times the value of the optimal solution to ν', as C has value at least as great as all coalitions in the optimal solution to ν'. Likewise, the sum of the values in ν'' of two disjoint coalitions, C_1 and C_2 that maximize $\nu''(C_1) + \nu''(C_2)$ will be at least $\frac{2}{m}$ times the value of the optimal solution to ν'', as C_1 and C_2 will have a combined value no less than the two highest valued coalitions in the optimal solution to ν''. We show how to extract the largest valued (two largest valued) coalitions from ν' (ν'') in $O(2^n)$ ($O(n2^n)$) time.

Our approximation technique employs the standard dynamic programming algorithm for coalition structure generation (Sandholm et al. 1999; Yeh 1986). The dynamic programming algorithm is based on the observation that if C is a coalition in an optimal solution CS to a subproblem $N' \subseteq N$, then $CS - \{C\}$ is an optimal solution to the subproblem $N' - C$. Observe that if there was a higher valued solution CS' to $N' - C$, then $CS' \cup \{C\}$ would be a better solution to the subproblem N' than CS, contradicting the fact that CS was an optimal solution to N'.

The dynamic programming algorithm works by iteratively constructing the optimal solution to each subproblem (subset of agents) in the order of increasing size. Prior to constructing the solution to a subproblem $N' \subseteq N$, the optimal solution for each subproblem of N' has to be determined. Let $opt_\nu(S)$ be the optimal solution to the subproblem S. That is, for each $S \subseteq N$, $opt_\nu(S)$ is a partitioning of S that maximizes:

$$\sum_{C \in opt_\nu(S)} \nu(C)$$

over all partitionings of S. When determining the optimal solution to the subproblem N', the dynamic programming algorithm considers each coalition in N' for possible inclusion in the solution to N'. Given a coalition $C \subseteq N'$, the highest valued solution to the subproblem N' to which C is a member is simply $\{C\} \cup opt_\nu(N' - C)$. Taking a maximum over all subsets of N' requires $O(2^{|N'|})$ time and results in the optimal solution to N'. Along with the value of the optimal solution to N', pointers are stored to both C and $N' - C$ (if the coalition N' is itself an optimal solution to the subproblem N' then a single pointer to itself is stored.) After all subproblems have been solved, the algorithm simply walks through the stored pointers to construct the optimal solution to the original problem.

The following definitions are used in the description of our algorithm.

Definition 1. *A characteristic function ν is* monotonic *iff for all $C, S \subseteq N$ such that $C \subseteq S$, $\nu(C) \leq \nu(S)$.*

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

Definition 2. *A characteristic function ν is k-*superadditive *iff for all $C, S \subseteq N$ such that*

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

then $\nu(C) + \nu(S) \leq \nu(C \cup S)$.

If ν is n-superadditive then it is superadditive. Superadditivity is a common assumption on the basis that if two disjoint coalitions C_1 and C_2 merge into $C_1 \cup C_2$, then at the very least the members of C_1 and C_2 can behave as if the merger did not take place and receive at least $\nu(C_1)$ and $\nu(C_2)$ respectively. Note that if ν is superadditive and non-negative (i.e., $\nu(C) \geq 0$, for all $C \subseteq N$), then it is monotonic; however, the reverse is not necessarily true. If ν is k-superadditive then for any $C \subseteq N$ such that $|C| \leq k$ and any partitioning P of C:

$$\nu(C) \geq \sum_{S \in P} \nu(S).$$

Our main result is now stated:

Theorem 1. *If ν is $\frac{n}{r}$-superadditive, then ν has an optimal solution that consists of at most $2r - 1$ coalitions.*

We make use of the following lemma in the proof of Theorem 1.

Lemma 1. *Let r and j be positive integers such that $0 \leq \frac{j}{r} \leq 1$ ($r \neq 0$) and let $x_1, \cdots, x_m$ be m arbitrary real numbers such that:*

1. $x_1 + x_2 + \cdots + x_m \leq \frac{j}{r}$
2. $\frac{1}{r} \geq x_1 \geq x_2 \geq ... \geq x_m > 0$

then $x_1, \cdots, x_m$ can be partitioned into $2j-1$ sets, $P_1, \cdots, P_{2j-1}$, such that the sum of the numbers in each set is at most $\frac{1}{r}$.

Proof of Lemma 1. For each $1 \leq i \leq j-1$, let $P_i = \{x_i\}$ and for each $j \leq i \leq 2j-1$ let:

$$P_i = \{x_k : k = i + lj \text{ for } l \geq 0\}.$$

For example, for $r = 3$, $j = 3$, and $m = 15$ the sets $P_1, \cdots, P_5$ are:

1. $P_1 = \{x_1\}$
2. $P_2 = \{x_2\}$
3. $P_3 = \{x_3, x_6, x_9, x_{12}, x_{15}\}$
4. $P_4 = \{x_4, x_7, x_{10}, x_{13}\}$
5. $P_5 = \{x_5, x_8, x_{11}, x_{14}\}$,

and for $r = 3$, $j = 2$ and $m = 10$ the sets $P_1, \cdots, P_3$ are:

1. $P_1 = \{x_1\}$
2. $P_2 = \{x_2, x_4, x_6, x_8, x_{10}\}$
3. $P_3 = \{x_3, x_5, x_7, x_9\}$.

For each P_i let V_i be the sum of the elements in P_i. Since each x_i is no more than $\frac{1}{r}$, clearly:

$$V_1, \cdots, V_{j-1} \leq \frac{1}{r}.$$

Note that, by construction, for $k > i \geq j$ we have the following:

1. $0 \leq |P_i| - |P_k| \leq 1$
2. $V_i \geq V_k$.

Thus, it suffices to show that $V_j \leq \frac{1}{r}$.

We proceed by way of contradiction. Thus, assume to the contrary that $V_j > \frac{1}{r}$.

Let $x_i \in P_j$. By construction, for each $0 < l < j$, $x_{i-l} \notin P_j$. Consider the sum:

$$\sum_{i=0}^{m} x_i = \sum_{x_i \in P_j} \left(x_i + \sum_{0<l<j} x_{i-l} \right) \geq j \cdot \sum_{x_i \in P_j} x_i > \frac{j}{r}.$$

This result contradicts the fact that the sum of the m numbers is less than $\frac{j}{r}$. Thus, $V_j \leq \frac{1}{r}$ and the lemma is true. □

We now prove Theorem 1.

Proof of Theorem 1. Let ν be an $\frac{n}{r}$-superadditive characteristic function. Let CS be any optimal coalition structure in ν. If CS contains $2r-1$ or fewer coalitions then the Theorem is true. Thus, assume that CS contains more than $2r-1$ coalitions. Let j be the number of coalitions in CS that consist of more than $\frac{n}{r}$ agents and let m be the number of coalitions in CS that contain no more than $\frac{n}{r}$ agents. Thus, $j < r$.

Let $C_1, \cdots, C_m$ be the m coalitions in CS that consist of no more than $\frac{n}{r}$ agents. Let $x_1, \cdots, x_m$ be the sizes of the m coalitions, represented as fractions of the total number of agents they contain (i.e., $x_i = \frac{|C_i|}{n}$). Assume w.l.o.g. that $x_1 \geq x_2 \geq \cdots \geq x_m$. Clearly, $x_1 + x_2 + \cdots + x_m \leq \frac{r-j}{r}$. By Lemma 1, $x_1, \cdots, x_m$ can be partitioned into $2(r-j)-1$ sets $P_1, \cdots, P_{2(r-j)-1}$ such that the sum of the numbers in each set P_i is no more than $\frac{1}{r}$. For each P_i, let P'_i be the set of agents defined by $P'_i = \bigcup_{x_l \in P_i} C_l$. Since the sum of the numbers in P_i is less than $\frac{1}{r}$, the number of agents in P'_i is less than $\frac{n}{r}$. Since ν is $\frac{n}{r}$-superadditive:

$$\nu(P'_i) \geq \sum_{x_l \in P'_i} \nu(C_l).$$

Thus, a new coalition structure CS' can be formed by substituting P'_i for $\{C_l : x_l \in P_i\}$ in CS that has value no less than CS. Since there are j coalitions consisting of greater than $\frac{n}{r}$ agents and the remaining coalitions consisting of no more than $\frac{n}{r}$ agents can be reduced to at most $2(r-j)-1$ coalitions, CS' consists of at most $2(r-j)-1+j = 2r-j-1$ coalitions. Since $j \geq 0$ the theorem follows. □

The following is an immediate corollary of Theorem 1:

Corollary 2. *If ν is $\frac{2n}{2r-1}$-superadditive, then ν has an optimal solution that consists of $2r-2$ coalitions.*

Proof. Since $\frac{2n}{2r-1} > \frac{n}{r}$, ν is also $\frac{n}{r}$-superadditive and by Theorem 1 has an optimal solution that consists of at most $2r-1$ coalitions. If ν contains an optimal solution on $2r-2$ or fewer coalitions, then the proof is complete. Thus, assume the smallest optimal solution, CS, in ν consists of exactly $2r-1$ coalitions.

Let C_1 and C_2 be the smallest two coalitions in CS and assume that $|C_1 \cup C_2| > \frac{2n}{2r-1}$. Thus, at least one of C_1 or C_2 must contain more than $\frac{n}{2r-1}$ agents. Since C_1 and C_2 are the two smallest coalitions in CS, all remaining coalitions must also contain more than $\frac{n}{2r-1}$ agents. As there are $2r-3$ coalitions in CS, other than C_1 and C_2, the total number of agents is bounded below by:

$$n > (2r-3) \cdot \frac{n}{2r-1} + \frac{2n}{2r-1} = n,$$

a contradiction. Thus $|C_1 \cup C_2| \leq \frac{2n}{2r-1}$. Define CS' as:

$$CS' = CS \cup \{C_1 \cup C_2\} - \{C_1, C_2\}.$$

Since ν is $\frac{2n}{2r-1}$-superadditive, $\nu(C_1 \cup C_2) \geq \nu(C_1) + \nu(C_2)$. Thus, $\nu(CS') = \nu(CS)$ (since CS was optimal). However, CS' contains one less coalition than CS, contradicting the assumption that the smallest optimal solution in ν consisted of $2r-1$ coalitions. □

Definition 3. *Let k and r be positive integers. Define $\nu_{\frac{k}{r}}$ as:*

$$\nu_{\frac{k}{r}}(C) = \begin{cases} \nu(opt_\nu(C)) & \text{if } |C| \leq \frac{kn}{r} \\ \nu(C) & \text{otherwise.} \end{cases}$$

The following theorem is an immediate observation.

Theorem 3. *$\nu_{\frac{k}{r}}$ is $\frac{kn}{r}$-superadditive.*

Proof. The theorem is clearly true, since for any $C, S \subseteq N$ such that $C \cap S = \emptyset$ and $|C \cup S| \leq \frac{kn}{r}$, $opt_\nu(C) \cup opt_\nu(S)$ is a coalition structure over $C \cup S$ with value $\nu_{\frac{k}{r}}(C) + \nu_{\frac{k}{r}}(S)$. □

Algorithm 1 Constructing $\nu_{\frac{k}{r}}$

```
for i = 1 to kn/r do
  for C ⊆ N, |C| = i do
    ν_{k/r}(C) ← ν(C)
    for C' ⊂ C do
      if ν_{k/r}(C') + ν_{k/r}(C − C') > ν_{k/r}(C) then
        ν_{k/r}(C) ← ν_{k/r}(C') + ν_{k/r}(C − C')
      end if
    end for
  end for
end for
ν_{k/r}(C) = ν(C) for all C such that |C| > kn/r
```

Algorithm 1 provides pseudo-code for the construction of $\nu_{\frac{k}{r}}$. Intuitively, Algorithm 1 runs the dynamic programming algorithm for only those coalitions consisting of $\frac{kn}{r}$ or fewer agents. For clarity, the code for maintaining pointers from each subproblem N' to subproblems C and $N' - C$ for constructing an optimal solution is not provided.

Note that every optimal solution to ν is also an optimal solution to $\nu_{\frac{k}{r}}$, since if C is in an optimal solution to ν then $opt_\nu(C) = \{C\}$. Given a solution $CS_{\nu_{\frac{k}{r}}}$ to $\nu_{\frac{k}{r}}$, a solution CS_ν to ν of equal value can be constructed as follows, assuming CS_ν is initially empty:

1. for $C \in CS_{\nu_{\frac{k}{r}}}$ s.t. $|C| > \frac{k}{r}$, let $CS_\nu \leftarrow CS_\nu \cup \{C\}$, and
2. for $C \in CS_{\nu_{\frac{k}{r}}}$ s.t. $|C| \leq \frac{k}{r}$, let $CS_\nu \leftarrow CS_\nu \cup opt_\nu(C)$.

The runtime of Algorithm 1 is provided in Theorem 4

Theorem 4. *Algorithm 1 computes $\nu_{\frac{k}{r}}$, for $k < \frac{r}{2}$ and $r \geq 2$, in:*

$$O\left(\frac{\sqrt{n} r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}} (r-k)^{\frac{(r-k)n}{r}}}\right)$$

time.

Proof. Algorithm 1 determines the optimal solution to each subproblem of N consisting of $\frac{kn}{r}$ or fewer agents. For a subproblem of size n', $2^{n'}$ time is required. The total runtime is bounded above by:

$$\sum_{0<i\leq\frac{kn}{r}} \binom{n}{i} 2^i \quad \leq \quad \frac{kn}{r}\binom{n}{\frac{kn}{r}} 2^{\frac{kn}{r}} = \frac{kn}{r} \frac{n! \cdot 2^{\frac{kn}{r}}}{\frac{kn}{r}!(n-\frac{kn}{r})!}.$$

Recall that Stirling's approximation states:

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

where e is the natural number.

Ignoring constant factors, the runtime of Algorithm 1 can be bounded from above by:

$$\begin{aligned} & \frac{kn}{r} \frac{n! \cdot 2^{\frac{kn}{r}}}{\frac{kn}{r}!(n-\frac{kn}{r})!} \\ = & \sqrt{\frac{k^2 n^3}{r^2 \frac{kn}{r}(n-\frac{kn}{r})}} \cdot \left(\frac{(\frac{n}{e})^n 2^{\frac{kn}{r}}}{\left(\frac{(\frac{kn}{r})}{e}\right)^{\frac{kn}{r}} \cdot \left(\frac{(n-\frac{kn}{r})}{e}\right)^{(n-\frac{kn}{r})}} \right) \\ = & \sqrt{\frac{kn}{(r-k)}} \cdot \left(\frac{n^n 2^{\frac{kn}{r}}}{(\frac{kn}{r})^{\frac{kn}{r}} \cdot (n-\frac{kn}{r})^{(n-\frac{kn}{r})}} \right) \\ = & \sqrt{\frac{kn}{(r-k)}} \cdot \left(\frac{r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}} (r-k)^{\frac{(r-k)n}{r}}} \right) \\ = & O\left(\frac{\sqrt{n} r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}} (r-k)^{\frac{(r-k)n}{r}}} \right), \end{aligned}$$

where the last equality holds since $k < \frac{r}{2}$. □

After $\nu_{\frac{1}{r}}$ is constructed, the highest valued coalition can be extracted in $O(2^n)$ time by looping through all subsets of N. Since $\nu_{\frac{1}{r}}$ has an optimal solution consisting of $2r-1$ or fewer coalitions, the highest valued coalition is at least $\frac{1}{2r-1}$ times the value of the optimal solution. A $\frac{1}{8}$ approximate solution can be obtained in $O(2^n)$ time by constructing $\nu_{\frac{2}{9}}$ and then extracting the highest valued coalition.

We now show how to extract two coalitions, C_1 and C_2, from an arbitrary ν that maximize $\nu(C_1)+\nu(C_2)$ in $O(n2^n)$.

Definition 4. *Define ν_{max} as:*

$$\nu_{max}(C) = \max_{S \subseteq C} \nu(S).$$

Intuitively, $\nu_{max}(C)$ represents the highest valued coalition that consists of only members in C. Algorithm 2 provides pseudo-code for constructing ν_{max} given ν. Algorithm 2 is based on the observation that $\max_{S\subseteq C} \nu(S)$ is either equal to $\nu(C)$ or is equal to $\max_{S\subseteq C-\{a\}} \nu(S)$ for some $a \in C$. Since Algorithm 2 constructs ν_{max} in the order of increasing coalition size, when computing $\nu_{max}(C)$ Algorithm 2 takes the maximum over $|C|+1$ values (i.e., $\nu(C)$ and $\nu_{max}(C-\{a\})$ for each $a \in C$).

During the construction of ν_{max}, for each $S \subseteq N$, in addition to storing the value of the largest subset of S in $\nu_{max}(S)$, a pointer from S to $\operatorname{argmax}_{C\subseteq S} \nu(C)$ is stored as well. Given a solution $CS_{\nu_{max}}$ to ν_{max}, a solution CS_ν to ν, of equal or greater value, can be constructed as follows. Assume CS_ν to be initially empty,

$$\text{for } C \in CS_{v_{max}}, \text{ let } CS_\nu \leftarrow CS_\nu \cup \{\operatorname*{argmax}_{S\subseteq C} \nu(S)\}.$$

Note that this procedure may not result in a partitioning of the agents (i.e., some agents may not appear in any of the

Algorithm 2 Constructing ν_{max}

```
for k = 1 to n do
    for C ⊆ N, |C| = k do
        ν_max(C) ← ν(C)
        for a ∈ C do
            if ν_max(C − {a}) > ν_max(C) then
                ν_max(C) ← ν_max(C − {a})
            end if
        end for
    end for
end for
```

coalitions in CS_ν). In this case, CS_ν can be extended to a partitioning of N by adding to CS_ν the coalition C that contains all agents that do not appear in some coalition in CS_ν.

For clarity pseudo-code for storing such pointers is omitted from Algorithm 2. The runtime of Algorithm 2 is provided in Theorem 5.

Theorem 5. *Algorithm 2 computes ν_{max} in $O(n2^n)$ time.*

Proof. See (Service and Adams 2010). □

Algorithm 3 provides pseudo-code for extracting two disjoint coalitions, C_1 and C_2, from N that maximize $\nu_{max}(C_1) + \nu_{max}(C_2)$. Algorithm 3 takes the maximum of $\nu_{max}(C) + \nu_{max}(N - C)$ for $C \subseteq N$. This results in an optimal partitioning of N into two sets. By the definition of ν_{max}, Algorithm 3 clearly results in an optimal partitioning of N into two sets. Since Algorithm 3 computes

$$\underset{C_1 \cap C_2 = \emptyset}{\operatorname{argmax}} (\nu_{max}(C_1) + \nu_{max}(C_2))$$

by a single loop through all subsets of N, it runs in $O(2^n)$ time.

Algorithm 3 Computing $\underset{C_1 \cap C_2 = \emptyset}{\operatorname{argmax}} (\nu_{max}(C_1) + \nu_{max}(C_2))$

```
C_1 = N
C_2 = ∅
for C ⊆ N do
    if ν_max(C)+ν_max(N−C) > ν_max(C_1)+ν_max(C_2) then
        C_1 = C
        C_2 = N − C
    end if
end for
```

Given ν, the overall approximation technique works as follows:

1. construct $\nu_{\frac{k}{r}}$ from ν, using Algorithm 1,
2. construct ν_{max} from $\nu_{\frac{k}{r}}$, using Algorithm 2,
3. extract the optimal partitioning of N into two sets from ν_{max}, using Algorithm 3,
4. convert the extracted solution to ν_{max} to a solution to $\nu_{\frac{k}{r}}$ and then to a solution to ν.

The runtime of this procedure is the maximum of $O\left(\frac{\sqrt{n} r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}} (r-k)^{\frac{(r-k)n}{r}}}\right)$ and $O(n2^n)$, the times required to compute $\nu_{\frac{k}{r}}$ and ν_{max}, respectively. However, this runtime can be reduced to the maximum of $O\left(\frac{\sqrt{n} r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}} (r-k)^{\frac{(r-k)n}{r}}}\right)$ and $O(2^n)$, at the cost of a reduced approximation ratio, by simply computing $\nu_{\frac{k}{r}}$ and extracting the highest valued coalition from $\nu_{\frac{k}{r}}$ by a single $O(2^n)$ time loop through all subsets of N.

For example, given a CFG ν, an approximate solution that is within $\frac{2}{3}$ of the optimal can be obtained in $O(\sqrt{n}2.83^n)$ by first constructing $\nu_{\frac{1}{2}}$ from ν, then constructing ν_{max} from $\nu_{\frac{1}{2}}$ and finally extracting the optimal partitioning of N into two sets from ν_{max} and converting that to a solution to ν. Similarly, a $\frac{1}{8}$ approximation solution to ν can be obtained in $O(2^n)$ time by constructing $\nu_{\frac{2}{9}}$, which requires less than $O(2^n)$ time, and then extracting the highest valued coalition from $\nu_{\frac{2}{9}}$ and converting it to a solution to ν.

Empirical Study

We compare our new approximate technique against the only other existing algorithm in the literature, our previous approximation technique (Service and Adams 2010), capable of guaranteeing constant factor approximation ratios in less than $O(3^n)$ time. The experiments incorporated problems consisting of 25 agents and are averaged over 20 independent runs. The performance of our new algorithm and our prior algorithm (referred to as Service and Adams' algorithm) are compared on two distributions:

1. normal distribution, the value of each coalition C is drawn from a normal distribution with mean $\frac{15}{4}$ and variance $\frac{1}{16}$.
2. modified uniform, 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.

Both algorithms are compared over a number of different approximation ratio guarantees.

Table 1 shows the performance of both algorithms on problems drawn from the modified uniform and normal distributions. While the performance of both algorithms greatly surpassed their respective theoretical guarantees, our algorithm generates the same performance guarantees, and the same empirical performance, far quicker than Service and Adams' algorithm. For example, in both cases our $\frac{1}{2}$ approximation ratio algorithm terminated in under 5 minutes, as opposed to Service and Adam's algorithm that required over 7 times longer. In the same amount of time required by Service and Adams' $\frac{1}{2}$ approximation ratio algorithm, our $\frac{2}{3}$ approximation ratio algorithm ran to completion.

Moreover, our $\frac{2}{3}$ factor algorithm always generated the optimal solution under the normal distribution and generated a solution, on average, within 99.5% of the optimal under the modified uniform distribution. Constructing the optimal solution with the standard dynamic programming algorithm required over 10 hours. This result shows that when time is

Table 1: The performances of the presented algorithm and Service and Adams' algorithm on the modified uniform and normal distribution, as a percentage of the optimal value. The time, in seconds, for each algorithm and each approximation ratio is given in parenthesis. Entries in the table are marked N/A for approximation ratios unobtainable by Service and Adam's algorithm.

	Approximation Ratio	$2/3$	$1/2$	$2/5$	$1/3$
Modified Uniform	Our Algorithm	99.5% (1773.6)	88.1% (246.0)	73.8% (37.3)	70.0% (24.3)
	Service and Adams' Algorithm	N/A	82.8% (1727.6)	N/A	72.1% (32.3)
Normal	Our Algorithm	100% (1748.1)	84.88% (245.3)	74.4% (37.0)	71.8% (24.1)
	Service and Adams' Algorithm	N/A	81.1% (1739.7)	N/A	74.4% (32.7)

an issue and a close to optimal solution is required, our algorithm is a significant practical improvement over the standard dynamic programming algorithm, as it is capable of generating near optimal solutions in an order of magnitude less time.

Conclusions

This paper presented a new algorithm for approximate coalition structure generation that provides better quality guarantees with lower worst case time than all known algorithms. Our approach is based on extracting approximate solutions from partially completed dynamic programming tables. We prove that after the standard dynamic programming algorithm for coalition structure generation has computed the optimal solution to all subproblems consisting of $\frac{n}{r}$ and fewer agents, then approximate solutions that are within a factor of $\frac{1}{r}$ and $\frac{2}{2r-1}$ of the optimal can be constructed in $O(2^n)$ and $O(n2^n)$ time, respectively. Our approach allows for constant factor approximations in less than $O(3^n)$ time. In particular, our approach can generate a $\frac{2}{3}$ approximate solution in $O(\sqrt{n}2.83^n)$ time and a $\frac{1}{8}$ approximate solution in $O(2^n)$ time.

Apart from the improvement over current existing constant factor approximation algorithms, we believe that Theorem 1 and its corollaries may be of independent interest. In particular, these results may prove useful in the analysis of other existing dynamic programming based algorithms or algorithms that use dynamic programming as a preprocessing phase, such as Rahwan and Jennings IDP-IP algorithm (Rahwan and Jennings 2008a).

Our approximation technique has the potential to be used as an anytime algorithm, that would be guaranteed to return the optimal solution in $O(3^n)$ time, similar to Service and Adams (Service and Adams 2010). The primary difference between the presented algorithm and Service and Adam's algorithm is that our algorithm incurs an overhead during the extraction of a solution rather than in a preprocessing phase. Thus, while guarantees on the quality of the solution are known at any point during the search, the actual value of the approximation solution will not be known until the extraction phase completes; making the determination of the exact value of the approximate solution capable of being extracted at each point in the search somewhat costly. Future work will determine the viability of the new approximation technique as an anytime algorithm.

References

Dang, V. D., and Jennings, N. 2004. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems*, 564–571.

Larson, K., and Sandholm, T. 2000. Anytime Coalition Structure Generation: An Average Case Study. *Journal of Experimental and Theoretical AI* 12:40–47.

Rahwan, T., and Jennings, N. 2008a. Coalition Structure Generation: Dynamic Programming Meets Anytime Optimisation. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, 156–161.

Rahwan, T., and Jennings, N. 2008b. An Improved Dyanmic Programming Algorithm for Coalition Structure Generation. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems*, 1417–1420.

Rahwan, T.; Ramchurn, S.; Jennings, N.; and Giovannucci, A. 2009. An Anytime Algorithm for Optimal Coalition Structure Generation. *Journal of Artificial Intelligence Research* 34:521–567.

Sandholm, T.; Larson, K.; Anderson, M.; Shehory, O.; and Tohmé, F. 1999. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence* 111(1-2):209–238.

Service, T. C., and Adams, J. A. 2010. Constant Factor Approximation Algorithms for Coalition Structure Generation. *Autonomous Agents and Multi-Agent Systems*. In Press.

Yeh, Y. 1986. A Dynamic Programming Approach to the Complete Set Partitioning Problem. *BIT Numerical Mathematics* 26(4):467–474.