Anytime Dynamic Programming for Coalition Structure Generation

(Extended Abstract)

Travis C. Service
Vanderbilt University
Nashville, Tennessee 37240, USA
tservice@acm.org

Julie A. Adams
Vanderbilt University
Nashville, Tennessee 37235-1824, USA
julie.a.adams@vanderbilt.edu

ABSTRACT

Determining the optimal coalition structure is a central problem in multi-agent systems. Two popular techniques include dynamic programming and anytime search algorithms. Dynamic programming algorithms guarantee an optimal solution and have the best worst case running time. Anytime algorithms are flexible as they can terminate before the search has completed, but have a significantly poorer worst case runtime. This paper provides an anytime dynamic programming algorithm with the worst case runtime of dynamic programming and the flexibility of anytime search.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms

Keywords

coalition formation, coalition structure generation

1. INTRODUCTION

Determining the optimal coalition structure is often studied in characteristic function games (CFGs). Given a set of agents A, a CFG is defined by a function $\nu : 2^A \rightarrow \mathcal{R}$, that assigns a value ν_C to each possible coalition C. The goal is to find a coalition structure, partitioning of the agents, CS^* that maximizes the sum of the values of its coalitions.

The majority of coalition structure generation algorithms are anytime algorithms that search the space of all possible coalition structures directly [3, 4]. These approaches are capable of generating high quality solutions, but in the worst case they examine all $O(n^n)$ coalition structures.

Rahwan and Jennings [2] advocated combining the worst case runtimes of dynamic programming with the flexibility of anytime search. They provide a parameterized combination of their anytime algorithm and their dynamic programming algorithm [2]. While their approach was empirically successful, the best known bound on its runtime is $O(n^n)$.

This paper introduces an anytime dynamic programming algorithm (ADP) with the worst case runtime of dynamic programming and the flexibility of anytime search. During preprocessing, ADP constructs a greedy solution. After preprocessing, an iterative algorithm solves incrementally larger subproblems exactly using dynamic programming. After r iterations, the generated solution is within a factor of $n - r$ of the optimal. If run to completion, the run time is $O(3^n)$.

Cite as: Anytime Dynamic Programming for Coalition Structure Generation (Extended Abstract), Travis C. Service and Julie A. Adams, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1411-1412

2. ANYTIME DYNAMIC PROGRAMMING

ADP employs two heuristics. A greedy solution is constructed and then iteratively larger subproblems are solved with dynamic programming. If run to completion, dynamic programming produces an optimal solution. If terminated prematurely, the better of the greedy solution and the current iterative solution is returned. It is counterintuitive that selecting the better of two heuristic solutions can produce guaranteed quality results. Lemma 1 allows such guarantees.

LEMMA 1. *Let CS^* and CS^*_{k+1} be optimal solutions to A and to a subproblem of A of $k+1$ agents and C^* be a maximum valued coalition then: $\nu_{CS^*} \leq (n-k) \cdot max(\nu_{C^*}, \nu_{CS^*_{k+1}})$.*

During preprocessing, an initial solution, CS_{greedy}, is constructed in $O(2^n)$ time by greedily adding the coalition of greatest value, out of the unassigned agents. By definition, the greedy solution contains the coalition C^*.

After preprocessing completes, solutions to incrementally larger subproblems are solved exactly with dynamic programming. The pseudo-code is provided in Algorithm 2.1. The quality guarantee follows directly from Lemma 1.

THEOREM 2. *After the solution to a subproblem of size $k+1$ is solved, the solution returned is guaranteed to be within a factor of $n-k$ of the optimal.*

If the iterative algorithm is run until a performance ratio of r is guaranteed, then the combined runtime of the preprocessing and iterative phases is $O(2^n + 3^{n-r})$. Thus, running the algorithm to completion requires $O(3^n)$ time, the same as the standard dynamic programming approach.

3. PERFORMANCE

ADP is compared with our previous algorithm [5] on two different problem distributions in a manner similar to previous empirical studies [1]. All experimental results are for 25 agents and are averaged over 25 runs.

Algorithm 2.1 Iterative Coalition Structure Generation

```
1: for k ∈ {2, ··· , n} do
2:   for i = 1 to |A_{n−k+1,n}| do
3:     for C ⊆ A_{n−k+1,n} : |C| = i and a_{n−k+1} ∈ C do
4:       f(C) ← max{f(C − C′) + f(C′) : C′ ⊂ C}
5:       C(C) ← C′ where C′ maximizes (4)
6:     end for
7:   end for
8: end for
9: CS* ← A_{n−k+1,n}
10: for C ∈ CS* do
11:   if C(C) ≠ C then
12:     CS*_{k+1} ← (CS* − C) ∪ {C(C), C − C(C)}
13:     Goto 10 and start with new CS
14:   end if
15:   CS_k ← best CS_{greedy} and CS*_{k+1} ∪ {{a_1}, ··· , {a_k}}.
16: end for
```

The performance of both algorithms under a uniform distribution was compared. The value of each coalition C was drawn uniformly from $[0, 10 \cdot |C|]$. Figure 1 shows the resulting performances. Our previous algorithm outperformed ADP for this problem distribution and found the optimal solution very quickly (on average less than 46 seconds). ADP slowly improved its approximation over time.

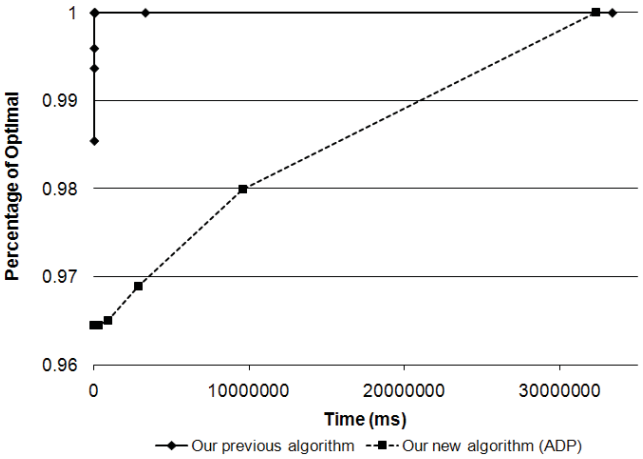

Figure 1: Performance of both algorithms as a function of time under a uniform problem distribution.

Both algorithms were also compared under a normal distribution. The value of each coalition C was drawn from a normal distribution with mean $\frac{15}{4}$ and variance $\frac{1}{16}$. Figure 2 shows the resulting performances. While our previous algorithm slowly improved its solution quality over time, ADP generated the optimal solution immediately after preprocessing terminated (approximately 7 seconds).

ADP's superior performance for the tested normal problem distribution appears to stem from the fact that, under the normal distribution, the optimal solution primarily consisted of the singleton coalitions. However, our previous algorithm was required to solve the entire problem to determine the optimal solution.

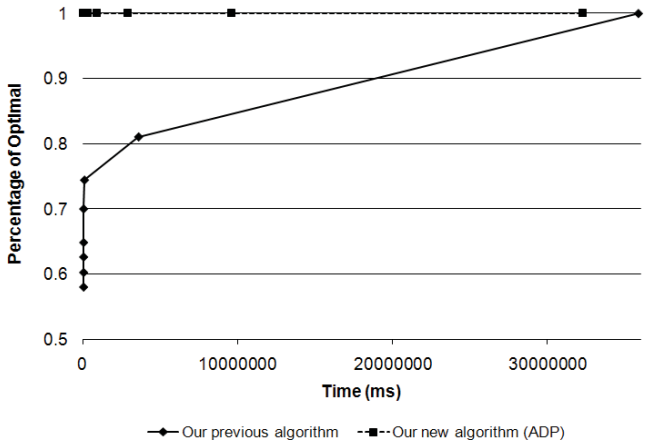

Figure 2: Performance of both algorithms as a function of time under a normal problem distribution.

4. CONCLUSIONS

This paper presented a new anytime algorithm for coalition structure generation and compared it with our previously developed algorithm [5]. Both algorithms are based on dynamic programming and employ different techniques for extracting approximate solutions from partially completed dynamic programming tables.

Both algorithms consist of a preprocessing phase followed by a dynamic programming algorithm. The preprocessing phase of ADP runs in $O(2^n)$ and the preprocessing phase of our previous algorithm runs in $O(n2^n)$. Time required for preprocessing large collections of agents can be a significant factor in deciding which algorithm to use.

While our previous algorithm is guaranteed to generate constant factor approximate solutions in less asymptotic time than ADP, the relative performances of the two algorithms under different problem distributions varies greatly.

5. REFERENCES

[1] Kate Larson and Tuomas Sandholm. Anytime Coalition Structure Generation: An Average Case Study. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 40–47, New York, NY, USA, 1999. ACM.

[2] Talal Rahwan and Nick Jennings. Coalition Structure Generation: Dynamic Programming Meets Anytime Optimisation. In *Proceedings of the 23rd Conference on Artificial Intelligence*, pages 156–161, 2008.

[3] Talal Rahwan, Sarvapali Ramchurn, Nicholas Jennings, and Andrea Giovannucci. An Anytime Algorithm for Optimal Coalition Structure Generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.

[4] Tuomas Sandholm, Kate Larson, Martin Anderson, Onn Shehory, and Fernando Tohmé. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[5] Travis C. Service and Julie A. Adams. Constant Factor Approximation Algorithms for Coalition Structure Generation. *Autonomous Agents and Multi-Agent Systems*, 2010, Accepted.