Heuristic methods for optimal coalition structure generation

Heuristic Methods for Optimal Coalition Structure Generation

Amir Hussin and Shaheen Fatima

Department of Computer Science
Loughborough University
Loughborough
United Kingdom

Abstract. The problem of finding the optimal coalition structure arises frequently in multiagent systems. Heuristic approaches for solving this problem are needed because of its computational complexity. This paper studies two such approaches: *tabu search* and *simulated annealing*. Through simulations we show that tabu search generates better quality solutions than simulated annealing for coalition games in *characteristic function form* and those in *partition function form*.

1 Introduction

In a multi-agent system, the agents work together and take cooperative actions to achieve complex tasks [18]. The effective formation of coalitions is therefore essential. Many situations require the formation of not just a single coalition but a coalition structure, i.e, an exhaustive partition of agents into non-overlapping coalititions. A primary challenge is to generate a coalition structure in which the entire system performance is maximised.

The optimal coalition structure generation problem is commonly modelled as a cooperative game in either *characteristic function form* or in *partition function form* [3]. The former are called characteristic function games (CFGs) and the latter partition function games (PFGs). In both CFGs and PFGs, the value of a coalition structure is given as the sum of the values of its coalitions. For CFGs, the value of a coalition is given in terms of its members. For PFGs, the value of a coalition depends not only on its members but also on how the external agents are organized. In other words, some *externalities* are inherent in PFGs.

For CFGs and more so for PFGs, finding an optimal coalition structure is computationally hard. A number of deterministic methods have been developed for PFGs [1, 9, 22] but they have exponential time complexity. This presents the need for developing effective heuristic methods for finding a good enough solution as quickly as possible, especially for games with a large number of agents. Such methods are important for example in mission critical systems where a group of agents representing emergency responders need to partition their resources so the emergency situation is handled optimally. In these systems the agents need to react quickly and time lost looking for the absolute optimal can severely impact on handling the emergency. A quick locally optimal solution would be better than a delayed globally optimal one because the situation may have changed during the time.

In the existing literature on heuristic methods, various approaches such as simulated annealing, genetic programming, particle swarm and ant colony have been studied for optimal coalition structure generation for CFGs (see Section 6 for details). However, the use of tabu search has so far not been studied even for CFGs although tabu search was previously shown to work well for other search problems such as bin packing [7], power systems network partitioning [4], crew scheduling [6], quadratic assignment [16], and vehicle routing [2]. Moreover, the use of heuristics for PFGs has been little explored in the literature.

Given this, we aim to compare two heuristic methods: *tabu search* and *simulated annealing* for finding an optimal coalition structure for both CFGs and PFGs. A key difference between tabu search and simulated annealing is that the former has a memory for recording the moves made during the search while the latter is memory-less and relies only on a random selection of next move. Since the latter requires less resource it might be more useful especially when there is a large number of agents provided the quality of its solution is good enough. Our objective is to determine how its performance compares to that of tabu search.

For evaluating the performance of a method, we consider the *quality of solution* generated and also the *time* taken to generate it. Since performance depends on the type of input data, we empirically measure average performance over a range of inputs. For this, we consider ten different probability distributions (see Section 5 for details) from which the input data is drawn randomly and then calculate average performance. Simulation results indicate that tabu search performs better than simulated annealing for CFGs and PFGs; given the same amount of time, tabu search yields a solution that is closer to the exact optimum than the solution generated by simulated annealing. Both tabu search and simulated annealing methods we implemented have *anytime* property; the quality of solution generated improves with the running time.

The main contributions of this paper are: i) it provides the first comparative evaluation of tabu search and simulated annealing for coalition structure generation, ii) the evaluation is done for both CFGs and PFGs, and ii) the evaluation is conducted extensively over a wide range of input data.

The paper is organised as follows. We begin in Section 2 by defining the problem. Section 3 is a description of tabu search and Section 4 of simulated annealing. Section 5 provides details of simulations set-up and a performance analysis of these two methods. Section 6 is about related literature and Section 7 draws conclusions.

2 Problem Specification

A coalitional game is a tuple $\langle A, v\rangle$ where $A = \{1,\dots,n\}$ is a set of n agents and v is a *coalition value function*. The definition of v depends on the type of coalition game. There are two types of coalition games: CFGs and PFGs.

Definition 1. *A* coalition structure *is an exhaustive partition of A into non-overlapping coalitions.*

In accordance with convention, the coalitions in any coalition structure will be arranged in the increasing order of their smallest members with the agents in the coalition being arranged in alphabetical order.

For example, for $A = \{1,2,3\}$, the 5 possible coalition structures are: $\{\{1\},\{2\},\{3\}\}$, $\{\{1,2\},\{3\}\}$, $\{\{1,3\},\{2\}\}$, $\{\{1\},\{2,3\}\}$, and $\{\{1,2,3\}\}$.

For n agents, there are $Bell(n) \sim \Theta(n^n)$ possible coalition structures, i.e, the number of coalition structures is exponential in n. Let Π^A denote the set of all possible coalition structures and $|\Pi^A|$ the cardinality of Π^A. We will first introduce CFGs and then PFGs.

CFGs: For these games, the value of a coalition depends solely on its members. v is a mapping of the form $v : 2^A \to \mathbb{R}$. It is a function that assigns a utility value to each subset, i.e., a *coalition* of A with $v(\phi) = 0$. All non-empty subsets of A are valid coalitions. A coalition is denoted by $C \subseteq A$ and the set of all possible coalitions as C^A. The cardinality of the this set is $|C^A| = 2^n - 1$.
Example 1: For $A = \{1,2,3\}$, v is defined as follows: $v(\{\phi\}) = 0$, $v(\{1\}) = 1$, $v(\{2\}) = 2$, $v(\{3\}) = 1$, $v(\{1,2\}) = 4$, $v(\{1,3\}) = 1$, $v(\{2,3\}) = 3$, $v(\{1,2,3\}) = 3$.

PFGs: For these games, the value of a coalition depends on its members and also on the coalition structure it is embedded in. v is a mapping of the form $v : 2^A \times \Pi^A \to \mathbb{R}$. It is a function that assigns a utility value to each pair comprised of i) a subset $C \subseteq A$, i.e., a *coalition* and ii) a partition in which C is embedded.
Example 2: For $A = \{1,2,3\}$, v is defined as follows: $v(\{1\},\{\{1\},\{2\},\{3\}\}) = 1$, $v(\{1\},\{\{1\},\{2,3\}\}) = 2$, $v(\{2\},\{\{1\},\{2\},\{3\}\}) = 1$, $v(\{2\},\{\{1,3\},\{2\}\}) = 4$, $v(\{3\},\{\{1\},\{2\},\{3\}\}) = 2$, $v(\{3\},\{\{1,2\},\{3\}\}) = 1$, $v(\{1,2\},\{\{1,2\},\{3\}\}) = 7$, $v(\{1,3\},\{\{1,3\},\{2\}\}) = 5$, $v(\{2,3\},\{\{1\},\{2,3\}\}) = 4$, $v(\{1,2,3\},\{\{1,2,3\}\}) = 3$.

The value of a structure CS is the sum of the values of its constituent coalitions. Thus we have:

$$v(CS) = \begin{cases} \sum\limits_{C \in CS} v(C) & \text{for CFGs} \\ \sum\limits_{C \in CS} v(C, CS) & \text{for PFGs} \end{cases}$$

Between all the $|\Pi^A|$ possible coalition structures, the problem is find one with the highest value, i.e., an *optimal coalition structure* which is defined as follows:

$$CS^* = \underset{CS \in \Pi^A}{arg\ max}\ v(CS) \tag{1}$$

Given the exponential number of possible structures, it is computationally infeasible to search the entire search space exhaustively. We therefore explore two heuristic search methods: *tabu search* and *simulated annealing*.

3 Heuristic search method 1: Tabu search

We begin with a quick overview of tabu search and then describe the method we implemented for optimal coalition structure generation. Tabu search [10] is a general heuristic method that attempts to quickly find high quality solutions by using a *neighbourhood search* guided with *tabu memory*. The basic concept is to maintain a tabu list of points already visited in the search space and avoid re-visiting them and those points which are known to be inferior to ones in the tabu list. By avoiding solutions that are already visited, the method ensures that new parts of the search space are being investigated. This also helps to avoid local maxima and thus enables better solutions to be found as the search progresses.

Algorithm 1 TACOS: Tabu search algorithm for finding an optimal coalition structure

```
1: tabuList ← [ ] {Tabu list initially empty}
2: CS ← randomly generated CS; {Generate a random start point}
3: currentBest ← CS
4: for iterationCount ← 1, maxIterations do {see Table 2 for maxIterations}
5:     Generate a neighbourhood N of CS
6:     Find the bestCS and the worstCS in N
7:     if worstCS not in tabuList then
8:         Add worstCS to tabuList
9:     end if
10:    if $v(bestCS) > v(currentBest)$ then
11:        currentBest ← bestCS
12:    end if
13:    if bestCS not in tabuList then
14:        add bestCS to tabuList
15:        CS ← bestCS
16:    else
17:        CS ← A randomly generated coalition structure that is not in tabuList
18:    end if
19: end for
20: Return currentBest
```

The tabu search method we implement is called TACOS (TAbu search for COalition Structure generation). The algorithm (see Algorithm 1) starts by constructing a random coalition structure as the initial starting point. The start point is the current best solution. This start point and its value are added as the first entry in the tabu list. The tabu list is an array that contains a list of coalition structures that is forbidden from being visited again as the search progresses. For the starting coalition structure, a neighbourhood is generated using *neighbourhood operators*. Four operators (explained in detail at the end of this section) were tested: *Split*, *Merge*, *Shift* and *Extract* to enable choosing the best combination. Between these four, the best combination was Shift and Extract.

Using Shift and Extract, a neighbourhood is generated and a *local maximum* and a *local minimum* are identified within it. The local minimum is added to the tabu list so that future iterations will rule out these candidates from being visited again (adding all the other solutions will result in a larger tabu list which will slow down the performance). If the local maximum is better than the current best, then the current best is updated.

During any iteration, if the local maximum is not in the tabu list, it is added to the list and search is continued from this point. On the other hand, if the local maximum is in the tabu list, it will not be explored as this point and its neighbourhood has already been visited. A random structure is then repeatedly generated until one that is not in the tabu list is found. This new structure becomes the continuation point for search.

The above process continues for a fixed number of iterations after which the current best solution is returned as the optimal one. It must be noted that the choice of neighbourhood operators (see Line 5 of Algorithm 1) is a key determinant of the performance of tabu search.

Neighbourhood generation: Although tabu search is a general method, the neighbourhood operators are problem specific. They must be defined such that the search space is explored effectively and repeated generation of the same coalition structures is avoided. Well designed neighbourhood operators are crucial to the success of tabu search. We defined the following four neighbourhood operators:

Merge The *immediate neighbour* $i_m(CS)$ of a coalition structure CS is the grand coalition if CS is the grand coalition. Otherwise, it is the structure obtained by merging the first two coalitions in CS. The neighbourhood N_m of CS is a set of coalition structures defined as follows:

$$N_m(CS) = i_m(CS) \cup i_m(i_m(CS)) \cup \ldots \cup i_m(GrandCoalition)$$

Extract The neighbourhood N_e of a coalition structure CS is a the coalition structure obtained by making all the agents in the largest coalition of CS singletons.

Split The *immediate neighbour* $i_s(CS)$ of a coalition structure CS is CS if CS is comprised of all singletons. Otherwise, it is the structure obtained by splitting the largest coalition in CS into two equal sized coalitions with split occuring in the middle. The neighbourhood N_s of CS is a set of coalition structures defined as follows:

$$N_s(CS) = i_s(CS) \cup i_s(i_s(CS)) \cup \ldots \cup i(AllSingletons)$$

Shift The *immediate neighbour* $i_{sh}(CS)$ of a coalition structure CS is CS if CS is the grand coalition. Otherwise, it is the structure obtained by moving the first agent from the second coalition of CS into the first coalition. The neighbourhood N_{sh} of CS is a set of coalition structures defined as follows:

$$N_{sh}(CS) = i_{sh}(CS) \cup i_{sh}(i_{sh}(CS)) \cup \ldots \cup i(GrandCoaliiton)$$

In order to study the efficacy of the above four operators, Algorithm 1 was tested for various combinations of these operators on ten different types of input data (details of input data are in Section 5). Between these combinations, the Shift and Extract combination yielded the best quality solutions. Thus for comparing the performance of TACOS with SA, the Shift and Extract combination was used.

4 Heuristic search method 2: Simulated annealing

Simulated annealing (SA) is a randomized local search method analogous to the metropolis algorithm [14]. It is based on the process of annealing in metallurgy [19]. As in gradient descent, this method iteratively generates random solutions. But in contrast to strict gradient descent, SA allows for a more extensive search for an optimal solution by accepting inferior solutions with some non-zero probability.

Algorithm 2 Simulated annealing algorithm for finding an optimal coalition structure

```
 1: $CS \leftarrow$ randomly generated CS; {Generate a random start point}
 2: $bestCS \leftarrow CS$
 3: $InitialTemperature \leftarrow 1.0$ {Initialize temperature}
 4: $\alpha \leftarrow 0.99$ {Initialize $\alpha$ used to update temperature}
 5: for $iterationCount \leftarrow 1, maxIterations$ do {see Table 2 for $maxIterations$}
 6:   Generate the neighbourhood $N$ of CS using Shift and Extract
 7:   $CS' \leftarrow$ A structure from $N$ chosen uniformly at random
 8:   if $v(CS') \geq v(CS)$ then
 9:     $CS \leftarrow CS'$ {Update $CS$}
10:   else
11:     $Probability \leftarrow e^{\frac{v(CS')-v(CS)}{t}}$ {Set probability of accepting an inferior solution}
12:     $CS \leftarrow CS'$ {Update $CS$}
13:   end if
14:   if $v(CS) \geq v(bestCS)$ then
15:     $bestCS \leftarrow CS$ {Update $bestCS$}
16:   end if
17:   $t \leftarrow t \times \alpha$ {Update $t$}
18: end for
19: Return $bestCS$
```

The SA method we implemented is given in Algorithm 2. The method starts with a randomly chosen start point called CS. During each iteration, a random neighbour CS' of CS is generated. The structure CS' is a randomly chosen structure from the neighbourhood N of CS generated using the combination of Shift and Extract operators (because as with TACOS, the Shift and Extract combination was found to be the best for SA). If the value of CS' is better, then CS' becomes CS. Otherwise, with probability $e^{\frac{v(CS')-v(CS)}{t}}$ CS' becomes CS. The temperature is updated. The process is repeated for a fixed number of iterations and the best solution found is returned as the optimal solution.

5 Performance Evaluation

The TACOS and simulated annealing methods were implemented in Python and their performance was evaluated in terms of two criteria: *solution quality* and *time* to generate the solution. Let CS_{TACOS} be the coalition structure returned by TACOS, CS_{SA} that for

Tabu search	Simulated annealing
Moves are random but recorded in tabu list to guide the search toward unexplored space	Moves are random and there is no record of moves
Advantage: Guided search and structured neighbourhood so only new solution space is explored avoiding repetition	Advantage: No neighbourhood structure, so search is quicker. Memory-less so requires less resource
Disadvantage: might cause a constrain on system resources	Disdvantage: Random selection could result in repeated moves and unguided jumps could leave the exploration stuck in worse parts of search space

Table 1. A comparison of tabu search and simulated annealing.

simulated annealing, and CS_{OPT} be the *exact* optimal solution. The solution quality for TACOS (and analogously for SA) was measured as follows:

$$\frac{v(CS_{TACOS})}{v(CS_{OPT})} \times 100 \tag{2}$$

For performance evaluation, the input data, i.e., the values of coalitions (see Equation 1) is generated from a wide range of probability distributions taken from the literature [24, 15]:

1. **Uniform (Standard)**: Python Mersenne twister pseudorandom number [13]: for all $C \in C^A$, the value $v(C) \sim U(0,1)$.
2. **Uniform (Sandholm)**: For all $C \in C^A$, the value $v(C) \sim U(0,|C|)$.
3. **Normal (Rahwan)**: For all $C \in C^A$, the value $v(C) \sim N(\mu,\sigma^2)$ where $\mu = 10 \times |C|$ and $\sigma = 1$.
4. **Exponential**: For all $C \in C^A$, the value $v(C) \sim |C| \times exp(\lambda)$, where $\lambda = 1$.
5. **Modified Uniform**: Each coalition's value is first drawn from $U(0,10 \times |C|)$, the value is then increased by a random number $r \sim U(0,50)$ with a probability of 0.2.
6. **Modified Normal**: Each coalition's value is first drawn from $N(0,10 \times |C|)$, the value is then increased by a random number $r \sim U(0,50)$ with a probability of 0.2.
7. **NDCS**: For all $C \in C^A$, the value $v(C) \sim N(\mu,\sigma^2)$, where $\mu = |C|$ and $\sigma = \sqrt{|C|}$.
8. **Beta**: For all $C \in C^A$, the value $v(C) \sim |C| \times Beta(\alpha,\beta)$, where $\alpha = \beta = 0.5$.
9. **Gamma**: For all $C \in C^A$, the value $v(C) \sim |C| \times Gamma(k,\theta)$, where $k = \theta = 2$.
10. **Agent-based Uniform**: Each agent a in a coalition C is a given a random power drawn from $\rho_a \sim U(0,10)$ to reflect its average contribution in all the coalitions it is a member of, then for every coalition C containing agent a, the actual power ρ_a^c of a in C is $\rho_a^c \sim U(0,2\rho_a)$. The value of $C \in C^A$ is the sum of the powers of all member coalitions. For all $C \in C^A$, $v(C) = \sum_{a \in C} \rho_a^C$.

For the above distributions, the quality of solution for each method was evaluated for up to 25 agents for CFGs and 9 agents for PFGs. Note that, although the heuristic methods can run for larger games, finding the exact optimum (needed for calculating

Probability distribution	CFGs (for 25 agents)			PFGs (for 9 agents)		
	Number of iterations		Avg. time	Number of iterations		Avg. time
	TACOS	SA	(ms)	TACOS	SA	(ms)
Uniform (Standard)	10000	20000	5500	25	75	9000
Uniform (Sandholm)	750	3000	400	250	375	90000
Normal (Rahwan)	50	150	30	10	20	3500
Exponential	200000	500000	120000	500	1000	180000
Modified Uniform	100000	200000	55000	500	1500	180000
Modified Normal	100000	200000	55000	500	1000	180000
NDCS	7500	30000	4000	500	1000	180000
Beta	600	2400	350	100	150	35000
Gamma	100000	350000	55000	100	250	35000
Agent-based Uniform	200000	400000	120000	500	1500	180000

Table 2. The number of iterations for each distribution. Time is in milliseconds (rounded to the next decimal).

the solution quality given in Equation 2) for them is computationally impractical. We present results for 25 agents for CFGs and 9 agents for PFGs. Both TACOS and the SA method we implemented are **oblivious to the type of probability distribution** from which the values of coalitions are drawn randomly. All simulations were run on a PC equipped with and Intel Xeon E5630 Processor running at 2.53Ghz (2.8Ghz Turbo) and 12GB RAM.

The performance of both TACOS and SA depends on i) the probability distribution from which the values of coalitions are drawn and ii) the random start point. Some probability distributions require more iterations than others to reach the same quality of solution. The number of iterations for each distribution was fixed based on a preliminary evaluation of TACOS and SA. The number of iterations is listed in Table 2. For each probability distribution, average performance was measured as follows:

CFGs The value of a coalition depends only on its members.

Step 1 : For a probability distribution, generate a data set comprised of all possible coalition structures and their associated values. A data set is generated as follows. For each possible coalition, randomly draw a value from the probability distribution. The value of a coalition structure is the sum of the values of its coalitions as given in Equation 1.

Step 2 : For a probability distribution, generate ten different data sets by repeating Step 1.

Step 3 : Repeat Step 2 for each of the ten probability distributions.

PFGs The value of a coalition depends on the structure it is embedded in.

Step 1 : For a probability distribution, generate a data set comprised of all possible coalition structures and their associated values. A data set is generated as follows. For each possible coalition structure, randomly draw a value from the probability distribution for each constituent coalition. The value of the structure is the sum of the values of its coalitions as given in Equation 1. The difference between CFGs and PFGs is that, for the latter, a random value for a coalition is drawn for each structure it is embedded in. But for the former, the value of a coalition is drawn only once. Thus, for PFGs, depending on the random values drawn, externalities may be *positive* or *negative* [5].

Step 2 : Repeat Step 2 for each of the ten probability distributions.

Since both TACOS and SA are sensitive to the start point which is random, these algorithms were run ten times for each data set. Average solution quality (with solution quality calculated as per Equation 2) and average running time were then measured across the ten runs for each data set and across the ten data sets for each probability distribution.

TACOS versus Simulated Annealing: The average performance of TACOS was compared with the average performance for SA for each of the ten probability distributions. The results are as shown in Figure 1 for CFGs and in Figure 2 for PFGs.

Consider the results in Figure 1 for CFGs. These results are for a system comprised of 25 agents. For each of the ten distributions, TACOS performed better than SA although the difference in performance varied form distribution to distribution. For the Normal (Rahwan) distribution, the performance of TACOS and SA was very close with the solution quality being 99.9% of the exact optimum. For the Beta and Uniform (Sandholm) distributions, TACOS achieved 99.9% and 98.7% respectively. The quality of solution was least for the Modified Normal and Modified Uniform distributions.

Consider the results in Figure 2 for PFGs. These results are for a system comprised of 9 agents. While running the TACOS and SA algorithms for more than 9 agents is easy, computing the exact optimum is computationally demanding. Although a 9 agent system is small, its analysis nevertheless helps in a comparative evaluation of TACOS and SA. For PFGs, TACOS generated better solutions than SA for each of the ten probability distributions. For the Gamma distribution, the heuristic solution for TACOS coincided with the exact optimum, for Normal (Rahwan) TACOS generated 99.96% of exact optimum while SA performed just fractionally below this level. For the Uniform (Standard), Beta, and Uniform (Sandholm) TACOS's solution was over 96% of the exact optimum. TACOS performed worst for the NDCS distribution achieving 80% of the exact.

Thus for both CFGs and PFGs, TACOS performed better than SA for each of the ten probability distributions. However, TACOS consumes more resource than SA in terms of its memory requirements.

Anytime property: A key feature of both TACOS and SA is that their average performance improves with execution time; the more the time spent running them, the better the results for CFGs and PFGs. This anytime property is illustrated in Figure 3 for TACOS for CFGs for each of the ten probability distributions.

6 Related Work

Existing methods for optimal coalition structure generation can be categorised into two types: *exact* and *approximate*. Exact algorithms return the absolute optimal solution by systematically exploring the space using methods such as dynamic programming to find an optimal one. Examples include [28, 23, 20]. Approximation methods that come with a guarantee on the quality of approximation include [26, 24, 1, 22, 9]. These methods in general have exponential running time (although polynomial time may be achieved by imposing restrictions on the coalitions that can form) which means that it is difficult for them to scale to large systems. Heuristic methods also generate approximate solutions but although they generally generate good solutions, there are no guarantees on the quality of solution. The advantage of these methods is that they typically have polynomial running time. The methods we explore in this paper belong to this class. In the remainder of this section, we will place our research in the context of existing heuristic methods that have previously been developed for coalition structure generation.

In their unpublished report Murillo et al. [17] used tabu search for set partitioning. The problem they addressed is to partition a set of objects so that similar items go into one class by assuming that the number of distinct classes is given (this is a special case of CFGs). They showed that tabu search performs better than simulated annealing and genetic algorithms. However, their simulations are limited to only five classes. There are two differences between their work and our paper. First, the problem we solve is much more general; we focus on determining the optimal coalition structure without knowing the number of partitions in the optimal structure. Thus, in their work, it is known that the optimal structure will contain, for example, five partitions. We find an optimal partition without knowing in advance how many coalitions that partition will contain. Second, they conducted simulations for a very restricted scenario in which the optimal partition contains at most five classes (i.e., coalitions). We consider CFGs of up to 25 agents and determine the optimal structure without knowing the number of coalitions in it. Third, they only considered CFGs while we considered both CFGs and PFGs.

Sen and Dutta [25] used genetic algorithms to find optimal coalition structures. They showed empirically that a good enough solution can be achieved. This work is an exception to other existing literature in that the simulations are conducted for CFGs and PFGs. They considered a particular search space for simulations by imposing a strict regularity on it. In contrast to this work, we do not impose any regularity on the search space. The simulations we conducted are comprehensive; we evaluated the average performance over 10 different probability distributions for the values of coalitions.

Heuristic techniques such simulated annealing [12], greedy search [8], particle swarm [11], and ant colony optimisation [27] have previously been used for CFGs. Ant colony and particle swarm optimization are decentralized methods in that a group of entities simultaneously search for an optimal solution. In contrast, simulated annealing and tabu search are centralized methods. See [21] for a detailed survey of coalition structure generation methods.

In summary, our aim in this paper is to focus on centralized approaches. To date, tabu search has not been used for coalition structure generation. Given this, we studied the efficacy of tabu search which is a memory-based method relative to simulated annealing which is a memory-less method. Note that, although simulated annealing was used in [12], its evaluation was limited in that its performance was evaluated for only one specific probability distribution where the value of any coalition is a random number between 0 and 1 was used. In contrast, our evaluation is comprehensive in that we evaluated and compared tabu search and simulated annealing for the ten different types of probability distributions (listed in Section 5). Furthermore, unlike previous work on heuristic methods, we consider both CFGs and PFGs.

7 Conclusions

In this paper, we implemented two heuristic methods, tabu search (TACOS) and simulated annealing and compared their average performance over a range of input data. Each of the two methods implemented is *oblivious to the type of probability distribution*. For both CFGs and PFGs, TACOS performed better than SA for each of the ten probability distributions. The price to pay for better performance is the extra memory requirement for TACOS. Both TACOS and SA have anytime property.

This paper evaluated the performance of TACOS and SA for small PFGs. Further work is needed to extend the results to larger PFGs. In addition, this paper used the same neighbourhood operators for CFGs and PFGs. A redefinition of operators particularly suitable for PFGs could improve performance. This too needs further research.

References

1. B. Banerjee and L. Kraemer. Coalition structure generation in multi-agent systems with mixed externalities. In *Proceedings of AAMAS*, pages 175–182, 2010.
2. G. Barbarosoglu and D. Ozgur. A tabu search algorithm for the vehicle routing problem. *Computers and Operations Research*, 26:255–270, 1999.
3. G. Chalkiadakis, E. Elkind, and M. Wooldrdidge. *Computational Aspects of Cooperative Game Theory*. Morgan and Claypool Publishers, 2011.
4. C. Chang, L. Lu, and F. Wen. Power system network partitioning using tabu search. *Electric Power Systems Research*, 49(1):55–61, 1999.
5. G. D. Clippel and R. Serrano. Margainal contributions and externalities in the value. *Econometrica*, 76(6):1413–1436, 2008.
6. T. Combs and J. Moore. A hybrid tabu search set partitioning approach to tanker crew scheduling. *Military Operations Research*, 9(1):43–56, 2004.
7. G. Crainic, G. Perboli, and R. Tadei. Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *Eur. J. Oper. Res*, 195:744–760, 2004.
8. N. DiMauro, T. Basile, S. Ferilli, and F. Esposito. Coalition structure generation with grasp. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 111–120. Springer, 2010.
9. A. Epstein and A. Bazzan. Distributed coalition structure generation with positive and negative externalities. In *Lecture Notes in Computer Science*, volume 8154, pages 408–419. Springer, 2013.
10. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
11. B. Guo and D. Wang. Optimal coalition structure based on particle swarm optimization algorithm in multi-agent system. In *Proceedings of the Sixth World Congress on Intelligent Control and Automation*, pages 2494–2497, 2006.
12. H. Keinanen. Simulated annealing for multi-agent coalition formation. In *Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 30–39, 2009.
13. M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, 1998.

14. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys*, 21:1087–1092, 1953.
15. T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, and N. R. Jennings. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, 230:14–50, 2016.
16. A. Misevicius. An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *OR Spectrum*, 34:665–690, 2012.
17. A. Murillo, E. Piza, and J. Trejos. A tabu search algorithm for partitioning. Technical report, 1999.
18. M.Wooldridge. *An introduction to multiagent systems*. John Wiley, 2009.
19. D. Pham and D. Karaboga. *Intelligent optimisation techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 2000.
20. T. Rahwan, T. Michalak, N. R. Jennings, M. Wooldridge, and P. McBurney. Coalition structure generation in multi-agent systems with positive and negative externalities. In *Proceedings of IJCAI*, pages 257–263, 2009.
21. T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings. Coalition structure generation: A survey. *AI Journal*, 229:139–174, 2015.
22. T. Rahwan, M. W. T. Michalak, and N. Jennings. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *AI Journal*, 186:95–122, 2012.
23. M. Rothkopf, A. Pekec, and R. Harstad. Computationally manageable combinational auctions. *Management Science*, 229:1131–1147, 1998.
24. T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Anytime coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999.
25. S. Sen and P. Dutta. Searching for optimal coalition structures. In *Proceedings of AAMAS*, pages 287–292, 2000.
26. O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *AI Journal*, 101(1–2):165–200, 1998.
27. A. Sukstrienwong. Searching optimal buyer coalition structure by ant colony optimization. *International Journal of Mathematics and Computers in Simulation*, 5:352–360, 2011.
28. D. Yeh. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26(4):467–474, 1986.

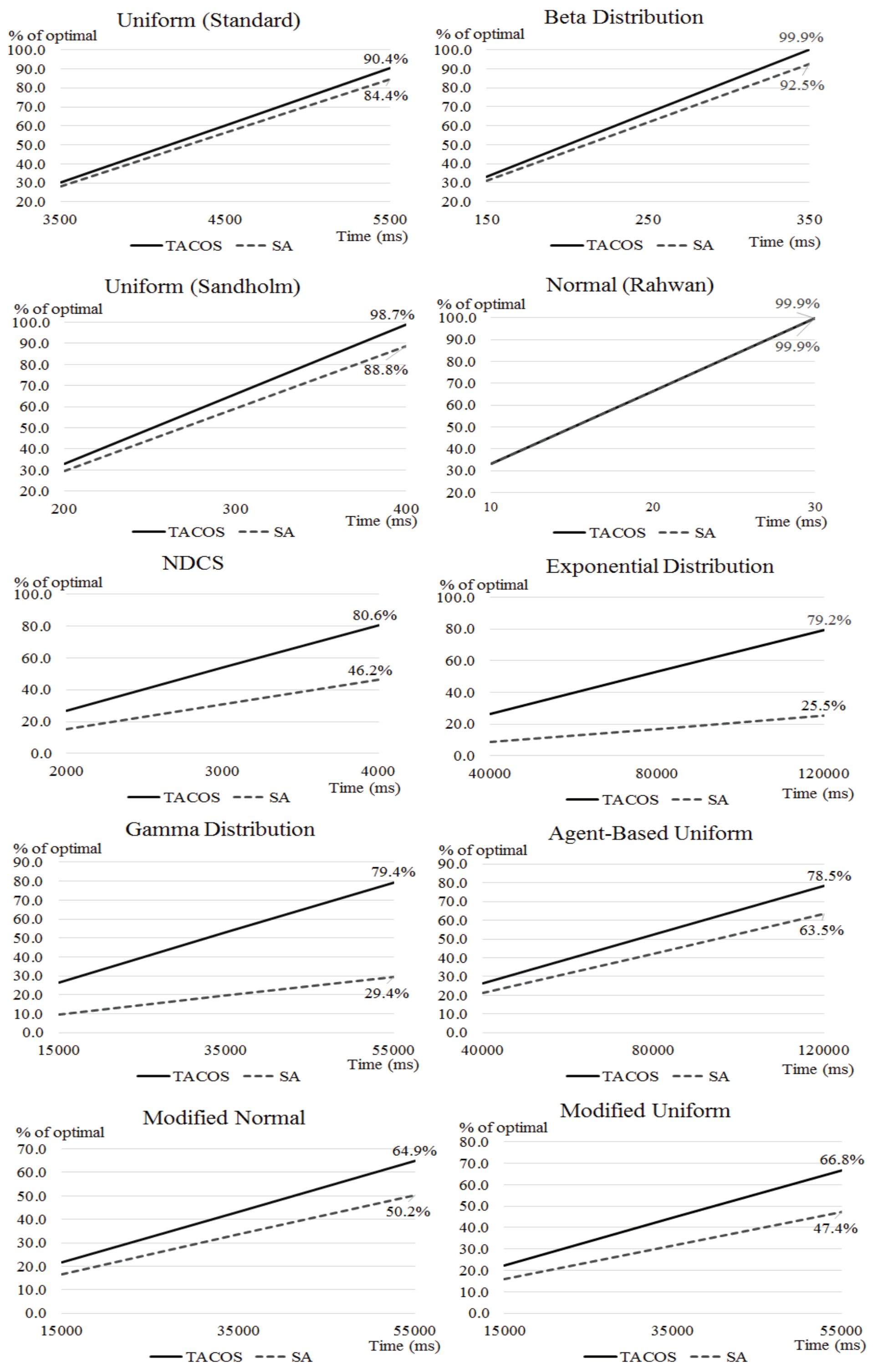

Fig. 1. A comparison of TACOS and simulated annealing for CFGs.

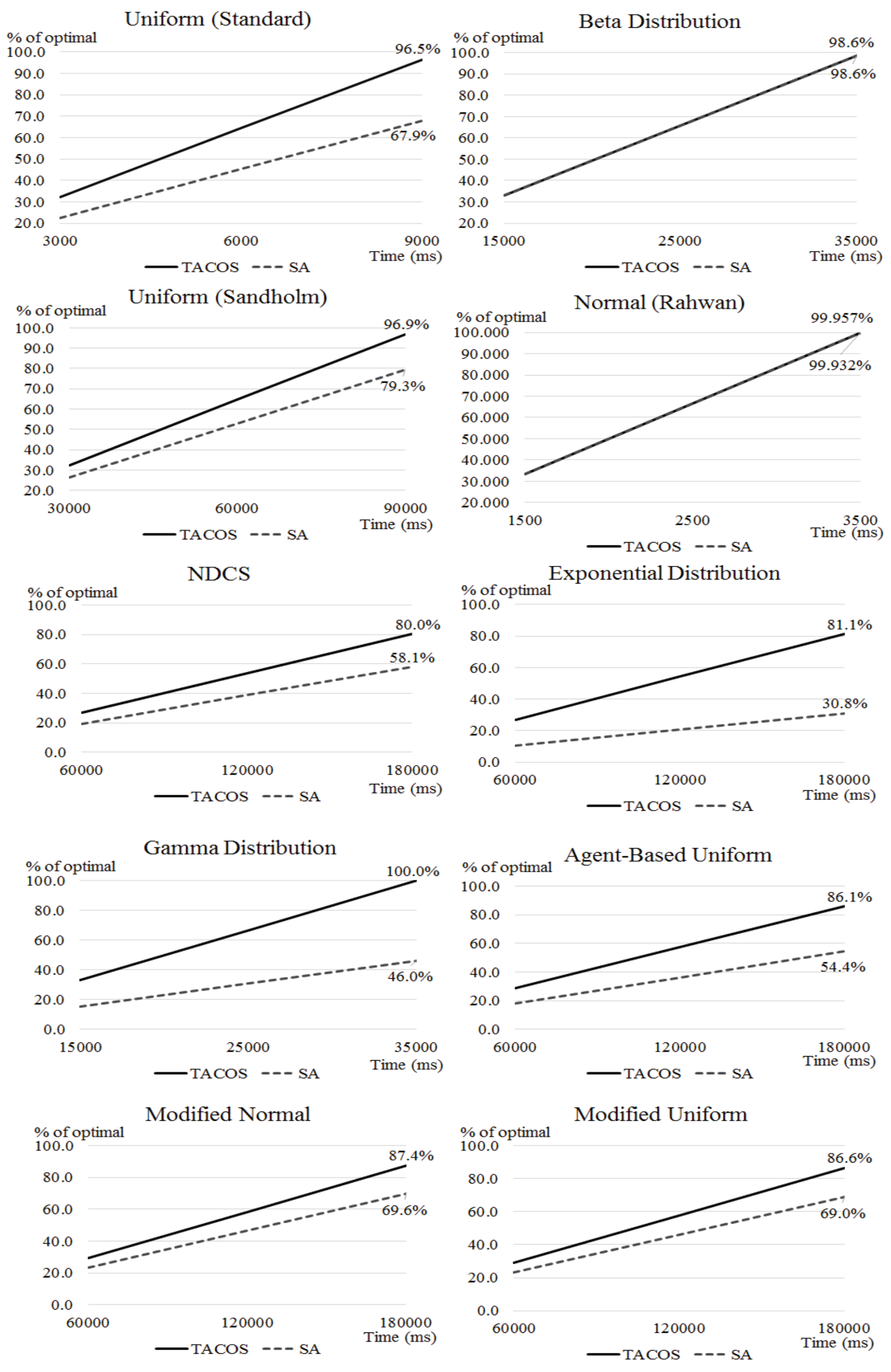

Fig. 2. A comparison of TACOS and simulated annealing for PFGs.

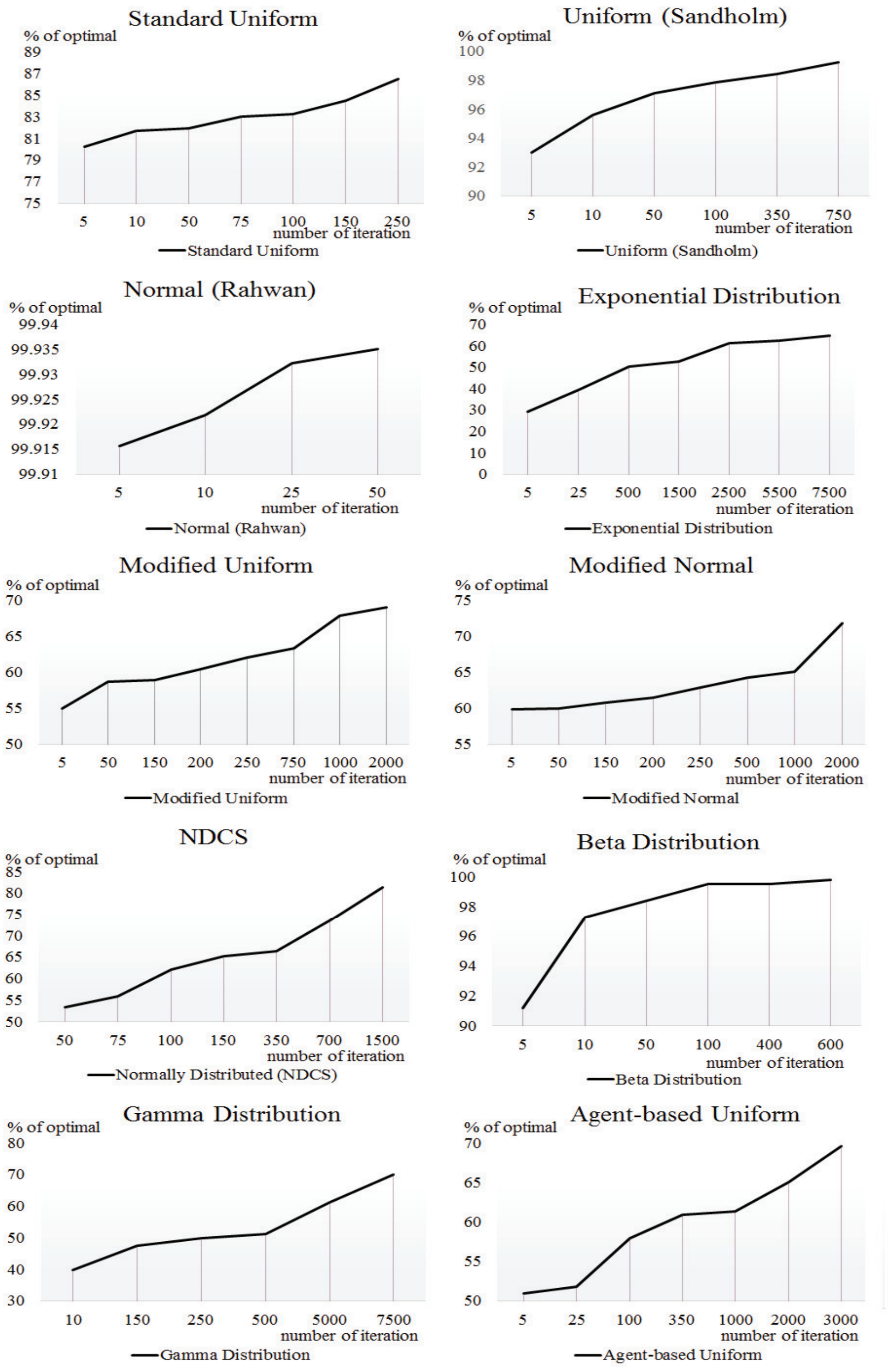

Fig. 3. An illustration of the **anytime** property of TACOS for CFGs.