Good friends are hard to find!

Thomas Brihaye
Institut de Mathématique – Univ. Mons-Hainaut – Belgium
thomas.brihaye@umh.ac.be

Mohamed Ghannem
ENSI – Univ. de la Manouba – Tunisia
ghannem.mohamed@yahoo.fr

Nicolas Markey
LSV – ENS Cachan & CNRS – France
markey@lsv.ens-cachan.fr

Lionel Rieg
Département Informatique – ENS Lyon – France
lionel.rieg@ens-lyon.fr

Abstract

We focus on the problem of finding (the size of) a minimal winning coalition in a multi-player game. We prove that deciding whether there is a winning coalition of size at most k is NP*-complete, while deciding whether k is the optimal size is* DP*-complete. We also study different variants of our original problem: the function problem, where the aim is to effectively compute the coalition; more succinct encoding of the game; and richer families of winning objectives.*

1. Introduction

Verification and control. Nowadays more and more real-life systems are controlled by computer programs. It is of a capital importance to know whether the programs governing these systems are correct. In order to formally verify them, those systems can be modeled by various mathematical models, such as *finite automata* [15] or *Kripke structures* [12]. Together with these models, several temporal logics have been considered, for instance LTL (linear-time temporal logic) [20] or CTL (computation-tree logic) [5, 21], in order to formally express "correctness". In this setting, given a model M and a temporal-logic formula φ, the *model-checking* problem asks whether M satisfies φ. Clearly if φ expresses a security specification of a system M, it is highly desirable to have *effective methods* for solving the corresponding model-checking problem. Several efficient model-checking tools have been developed and applied with great success over an abundant number of industrial case studies [16, 4, 9].

Model-checking focuses on *closed systems*, *i.e.*, systems where all the actions are controlled by a single agent. If we want to distinguish between actions of a *controller* and actions of an *(hostile) environment*, or more generally if we are interested in modelling interactions between several agents, we have to consider *open systems*. These systems are studied in *control theory* [2, 22], where the ultimate goal is to automatically synthesize a controller that will restrict the behavior of the system in order to satisfy some given properties. The natural mathematical framework to discuss open systems is *game theory*. A natural game version of finite automata and Kripke structures is the model of *concurrent game structure* (CGS) [1]. CGSs are finite-state automata whose transitions conform to the following protocol: at each step, all the players select one of the moves they are allowed to play, and the next state is looked up in the transition table of the CGS. As regards specification languages, alternating-time temporal logic (ATL) has been proposed in [1] as an extension of CTL for reasoning about strategies: path quantifiers are replaced with *strategy quantifiers*. It is then possible to express controllability properties, such as "Player 1 has a strategy to reach some given state". Compared to CTL, this extension comes with no extra cost: it can still be verified in polynomial time [1] (though there can be slight discrepancies when the number of agents is a parameter [14]).

Our contribution. In this paper, we study a different facet of ATL. Instead of checking if an ATL formula holds on a given CGS, we rather consider the question of finding a minimal coalition allowing to reach an objective. For example, when modelling a communication network, we could wonder how many nodes of the network have to be controlled in order to ensure that a message can transit towards its destination. Our results easily extend to the case where agents have a "price": in the example above, some nodes might be more difficult than others to control, which could be expressed as the *cost* of controlling them. The problem would then be to find the cheapest coalition that can enforce our goal.

This kind of problems can be naturally encoded in our

This work was done while the first, second and fourth authors were visiting LSV. It was partly supported by project DOTS (ANR-06-SETI-003).

setting, and we settle the precise complexity of this and several related problems: for simple winning objectives, deciding whether there exists a winning coalition of size k is NP-complete, while deciding whether k is the optimal size is DP-complete. Furthermore, we study different variants of our original problem: the function problem, where the aim is to effectively compute the coalition, more succinct encoding of the game, and richer families of winning objectives.

Related work. Coalition formation is an active topic in cooperative game theory [17] and has recently been extended to the multi-agent setting, leading to interesting applications (see e.g. [23, 6, 7]). In particular, qualitative coalition games are studied in [25], where the authors end up with similar unusual complexity classes as ours. Still, the problems they consider are quite different from ours, and we see no direct relationship between them (even in the light of [8]).

Our optimal coalition problem is in some sense related to the *temporal logic query problem* introduced in [3] and further studied in [10]. Our algorithm could be used to solve ATL queries of the form $\langle\langle ? \rangle\rangle\, \phi$.

Outline of the paper. We introduce the necessary definitions in Section 2. In Section 3, we fully solve our problems in the case of reachability objectives in explicit CGSs, proving their NP-completeness and DP-completeness. Last, Section 4 presents several extensions of this result to symbolic (or "implicit") CGSs [14], richer winning objectives and DCGS [13]. Since some of the complexity classes we use are non-standard, we also include some related definitions in Appendix A.

2. Definitions

Definition 1 ([1])**.** *A* Concurrent Game Structure *(*CGS *for short) $\mathcal{C}$ is a* 6*-tuple* $(\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathsf{Mv}, \mathsf{Edg})$ *where:*

- $\mathsf{Agt} = \{A_1, ..., A_k\}$ *is a finite set of* agents *(or* players*);*
- Loc *and* AP *are two finite sets of* locations *and* atomic propositions*, resp.;*
- $\mathsf{Lab}\colon \mathsf{Loc} \to 2^{\mathsf{AP}}$ *is a function labeling each location by the set of atomic propositions that hold for that location;*
- $\mathsf{Mv}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathbb{N}) \setminus \{\varnothing\}$ *defines the (finite) set of possible moves for each agent in each location.*
- $\mathsf{Edg}\colon \mathsf{Loc} \times \mathbb{N}^k \to \mathsf{Loc}$*, where* $k = |\mathsf{Agt}|$*, is a (partial) function defining the transition table. With each location and each set of moves of the agents, it associates the resulting location.*

The intended behavior of such a model is as follows: in any location q, each player A_i independently chooses one of its allowed moves m_i (defined by $\mathsf{Mv}(q, A_i)$). Once they all have chosen, the execution switches to the location given by $\mathsf{Edg}(q, (m_i)_{A_i \in \mathsf{Agt}})$. Formally, given a location q, a coalition $A \subseteq \mathsf{Agt}$, and a set of moves $(m_i)_{a_i \in A}$ for the players in A, we let $\mathrm{Next}(q, (m_i)_{A_i \in A})$ be the set $\{\mathsf{Edg}(q, (m_i)_{A_i \in \mathsf{Agt}}) \mid \forall j \notin A.\ m_j \in \mathsf{Mv}(q, A_j)\}$. We write $\mathrm{Next}(q)$ for the set of possible successors of location q (*i.e.*, $\mathrm{Next}(q, \varnothing)$).

Definition 2. *An* execution *of a CGS $\mathcal{C}$ from location q_0 is an infinite sequence $(q_i)_{i \in \mathbb{N}}$ s.t., for any $i > 0$, $q_i \in \mathit{Next}(q_{i-1})$.*

Remark. *Complexity results heavily rely on how the transition table* Edg *is encoded. Following [14], we consider two ways of encoding the table:*

- *the first way is to explicitly list the cells of the table. If each of the k agents has two possible moves, this ends up in a table of size $2^k \cdot |\mathsf{Loc}|$.*
- *the second way is by a finite list of pairs $(\phi_i, q_i)_{i \in \mathbb{N}}$, where each q_i is a location in* Loc*, and ϕ_i are positive boolean combinations of propositions of the form $m_i = c$, whose individual meaning is "player A_i plays move c". The first formula that holds true indicates the successor state. For example, given the list*

$$([m_1 = 1, q_1], [m_3 = 2, q_2], [\top, q_3]),$$

the play goes to location q_1 if player 1 chooses move 1, otherwise it goes to location q_2 if player 3 chooses move 2, otherwise it goes to q_3. W.l.o.g., we require that the last formula always be $\top$.

CGSs with the first encoding are called explicit CGSs*, while the second type of CGSs are* symbolic CGSs*. It is easy to figure out how to translate one into the other, the translation to explicit CGSs being exponential.*

Definition 3. *Let $A_i \in \mathsf{Agt}$ be a player. A* strategy *for player A_i is a total function $f_i\colon \mathsf{Loc}^{(\mathbb{N})} \to \mathbb{N}$ mapping each prefix of an execution to an integer. It is required that the result of a strategy is a possible move for the player in the last location of the prefix: $\forall \rho = q_0\, q_1\, \ldots\, q_k.\ f_i(\rho) \in \mathsf{Mv}(q_k, A_i)$.*

Given a coalition $A \subseteq \mathsf{Agt}$, a strategy *for coalition A is a sequence of strategies $(f_i)_{A_i \in A}$, one for each player in A.*

With the definitions above, it is easy to define the outcomes of a strategy:

Definition 4. *Let $(f_i)_{A_i \in A}$ be a strategy for a given coalition $A \subseteq \mathsf{Agt}$. An execution $\rho = q_0\, q_1\, \ldots$ of $\mathcal{C}$ is an* outcome *of the strategy $(f_i)_{A_i \in A}$ if*

$$\forall j > 0.\ q_j \in \mathit{Next}(q_{j-1}, (f_i(q_0\, \ldots\, q_{j-1}))_{A_i \in A}).$$

The set of outcomes of $(f_i)_{A_i \in A}$ *from* q_0 *is denoted by* $\mathit{Out}(q_0, (f_i)_{A_i \in A})$.

Definition 5. *A* winning objective Ω *is a set of executions. A strategy* $(f_i)_{A_i \in A}$ *is winning from location* q *if* $\mathit{Out}(q, (f_i)_{A_i \in A}) \subseteq \Omega$.

In the rest of the paper, we will mainly focus on *reachability* objectives: given a set of goal locations $F \subseteq \mathsf{Loc}$, the winning objective consisting in reaching F is defined as $\Omega_{\mathrm{Reach}(F)} = \{(q_i)_{i \in \mathbb{N}} \mid \exists i.\ q_i \in F\}$. Other objectives such as *safety*, *Büchi* or LTL objectives can be defined in a similar way.

In the sequel, we focus on the problem of finding (the size of) an optimal coalition (in terms of its size) having a winning strategy:

Definition 6. *The problem of* coalition size *is defined as follows: given a CGS* $\mathcal{C}$*, an initial location* $q_0 \in \mathsf{Loc}$*, a winning objective* Ω*, a maximal size* $k \in \mathbb{N}$*; determine if there exists a coalition of size at most* k *having a winning strategy from* q_0 *for the objective* Ω.

We also define the following variants:

Definition 7. *Given a CGS* $\mathcal{C}$*, a coalition* $A \subseteq \mathsf{Agt}$*, a location* $q_0 \in \mathsf{Loc}$*, a winning objective* Ω*, and an integer* $k \in \mathbb{N}$*, we define the following problems:*

- *the problem of* constrained[1] *(resp.* co-constrained*)* coalition size *consists of determining if there exists a coalition of size at most* k*, not containing* A *(resp. containing* A*), having a winning strategy from* q_0 *for the objective* Ω.
- *the problem of* optimal coalition size *consists of determining if there exists a coalition of size exactly* k *having a winning strategy from* q_0 *for the objective* Ω*, and no smaller coalition has.*

The next section focuses on those four problems in the case of explicit CGSs, for reachability objectives. The other combinations are explained in Section 4.

3. Optimal coalition problems for reachability objectives on explicit CGSs

It is well-known that, given a coalition in an *explicit* CGS, deciding if it has a winning strategy for reachability objectives can be done in polynomial time [1, 14]. This yields an NP algorithm for the coalition size problems, consisting in non-deterministically guessing a candidate coalition, and checking that it has a winning strategy.

As a consequence, the optimal coalition size problem is in DP[2]: the optimal coalition size is k iff there is a winning coalition of size at most k, and no winning coalition of size at most $k-1$.

We prove that the algorithms above are optimal (even on turn-based[3] games).

Theorem 8. *The ((co-)constrained) coalition size problem with reachability objectives is* NP*-hard on explicit CGSs.*

Proof. We begin with the constrained version, reducing 3SAT to it[4]. Let $\phi = \bigwedge_{1 \leq i \leq n}(l_{i,1} \vee l_{i,2} \vee l_{i,3})$ be an instance of 3SAT on m variables x_1 to x_m. We build a turn-based game with $3m + n + 3$ states, each one being controlled by a different player, where:

- there are two states labeled 0 and 1, the second one being the goal to be reached. There is a self-loop on each of these two states (this is only because our definitions require each location to have a successor);
- for each variable x_j, with $1 \leq j \leq m$, there are three states labeled x_j, $\neg\, x_j$ and $x_j?$. The states x_j and $\neg\, x_j$ have transitions to both states 0 and 1, while state $x_j?$ has three transitions to x_j, $\neg\, x_j$ and 0;
- for each clause $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, with $1 \leq i \leq n$, there is one state, with four transitions: one to each of the three states corresponding to literals $l_{i,1}$ to $l_{i,3}$, and an additional one to state 0;
- last, the initial state q_0 has $n + m$ transitions, one to each "clause"-location c_i and one to each "question-mark" location $x_j?$.

For the sake of simplicity, we identify each player with the name of the state it controls. Fig. 1 depicts the resulting turn-based game (where a solid transition from/to a set of nodes indicates a transition from/to each node of the set; dashed transitions represent transition from a clause node to each literal node it contains).

Our instance of the constrained coalition size problem is defined on the structure above, with the following additional requirements: we are looking for a coalition of size at most $n + 2m$ not containing player q_0.

First, assume that our initial instance ϕ of 3SAT is satisfiable. Let v be a valuation of the variables x_1 to x_n satisfying ϕ. Consider the coalition containing the n "clause" players, the m "question-mark" players, and, for each $1 \leq j \leq m$, player x_j if $v(x_j) = \top$ and player $\neg\, x_j$ if $v(x_j) = \bot$.

[1] Since our algorithms consist in guessing a candidate coalition and checking that it is winning, we could have much more general constraints on coalitions, provided that they can be checked in polynomial time.

[2] See Appendix A for some notes about this and other complexity classes.

[3] A game is *turn-based* if, in each location, all but one player have only one possible move.

[4] Other reductions, *e.g.* from VERTEX-COVER, could also be achieved. Still, the reduction we present here is the most elegant we found that yields a turn-based game and that can be used for our extensions of Section 4.2.

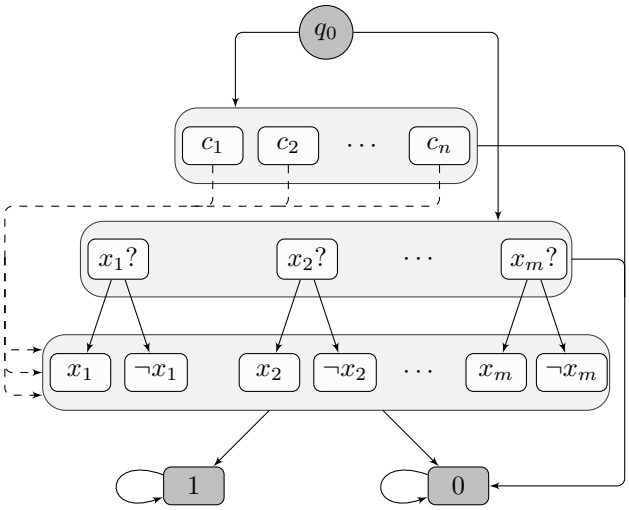

Figure 1. Global schema of the turn-based game used for the reduction.

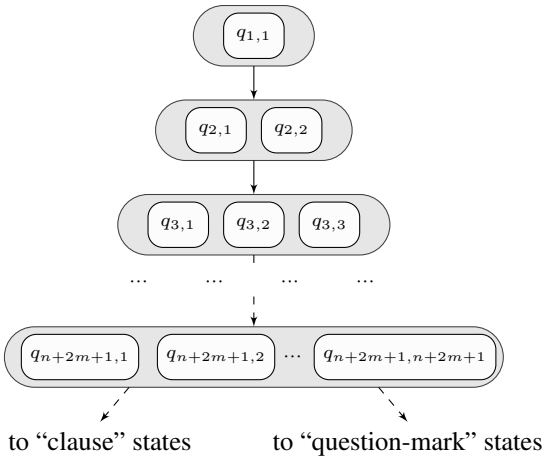

Figure 2. The module replacing q_0.

This coalition has size $n + 2m$, and does not contain q_0. It remains to exhibit a winning strategy for this coalition:

- since each clause is made true by valuation v, (at least) one of the successors of each "clause" state belongs to the coalition. The strategy of each "clause" player is to go to that state;
- similarly, each "question-mark" player will send the play to the (only) successor that belongs to the coalition;
- last, the "literal" players will jump to the goal state 1.

It is clear enough that this strategy is winning: player q_0 cannot do anything but go to a state that belongs to the coalition. Then, the strategy of the "clause" player and of the "question-mark" player sends the play to a "literal" player that belongs to the coalition, which herself goes to the goal location.

Conversely, assume that there is a winning coalition of size at most $n + 2m$ not containing q_0. First notice that any winning coalition necessarily contains the "clause"- and "question-mark" players (otherwise, the coalition has no winning strategy). For the same reason, for each variable x_j, at least one of x_j and $\neg\, x_j$ must be in the coalition.

This makes $n + 2m$ players, so that our winning coalition contains no other player. As a consequence, for each $1 \leq j \leq m$, exactly one of x_j and $\neg\, x_j$ belongs to the coalition. This defines a valuation v, with $v(x_j) = \top$ iff player x_j belongs to the winning coalition.

It remains to prove that this valuation satisfies ϕ. Since the coalition is winning, each "clause" player can send the play in a state that belongs to the coalition. By construction, this implies that the corresponding clause is satisfied under valuation v.

We now extend this reduction to the non-constrained problem. In order to relax the constraint that player q_0 does not belong to the coalition, we add several "copies" of q_0 (and as many new players) in the following way: for $1 \leq k \leq n + 2m + 1$, and for each $1 \leq k' \leq k$, there is a state $q_{k,k'}$. Each state $q_{k,k'}$, with $1 \leq k < n + 2m + 1$, has $k + 1$ transitions, to states $q_{k+1,1}$ to $q_{k+1,k+1}$. States $q_{n+2m+1,1}$ to $q_{n+2m+1,n+2m+1}$ have the same transitions as q_0 in the previous reduction. The new initial state is $q_{1,1}$. This construction is depicted on Fig. 2. The rest of the construction is similar to the previous one.

As in the constrained case, if our initial instance of 3SAT is positive, we can build a coalition of size $n + 2m$ and its strategy, and prove that this strategy is winning: after the first $n + 2m + 1$ steps, the play will end up in a ("clause" or "question-mark") state controlled by the coalition, which will win by applying its strategy.

Conversely, if there is a winning coalition of size at most $n+2m$, this coalition contains none of the newly added players. Indeed, if a coalition A of size (at most) $n + 2m$ is winning and contains some player $q_{k,k'}$, then the coalition A' obtained from A by removing $q_{k,k'}$ is also winning: this is because whatever the strategy for coalition A, there is an outcome ending up in an uncontrolled state $q_{n+2m+1,l}$. Since coalition A has a winning strategy from that uncontrolled state, so does coalition A'. Also, by symmetry of the roles of the states $q_{n+2m+1,l}$ in the lower set, coalition A' has a strategy from any of those states, thus also from state $q_{1,1}$. As a consequence, if there is a winning coalition of size at most $n + 2m$, then there is one containing none of the states $q_{k,k'}$. We are then back to the situation of the previous reduction.

As for the co-constrained problem, it contains as a special case the unconstrained problem, and is thus also NP-hard. □

Theorem 9. *The optimal coalition size problem for reachability objectives is* DP*-hard on explicit CGSs.*

Proof. Given a pair of boolean formulas (ϕ, ϕ') in 3-CNF, the problem SAT-UNSAT returns "yes" iff ϕ is satisfiable and ϕ' is not. This problem is easily shown DP-complete [18]. We reduce this problem to the (constrained) optimal coalition size problem, following the lines of the proof for NP-hardness. Pick an instance (ϕ, ϕ'), where $\phi^{(\prime)} = \bigwedge_{1 \leq i \leq n^{(\prime)}} (l_{i,1}^{(\prime)} \vee l_{i,2}^{(\prime)} \vee l_{i,3}^{(\prime)})$ (assumed w.l.o.g. to involve disjoint sets of variables) of SAT-UNSAT. We first apply some transformations to ϕ and ϕ':

- following the proof of [18, Theorem 17.2], we transform ϕ' so that at least all but one clause is satisfiable: this is achieved by disjuncting a fresh variable z' to each clause, and adding the extra clause $\neg z'$. The resulting formula is satisfiable iff the original one was, and by setting all variables to true, all but one clause is satisfied. In the sequel we denote by ϕ' the transformed formula and assume it contains n' clauses and m' variables (including the variable z').
- as regards ϕ, we duplicate each variable and each clause and the new formula is the conjunction of ϕ and $\hat{\phi}$, where $\hat{\phi}$ is obtained by replacing each variable x_i with $\hat{x}_i$. Hence if ϕ is not satisfiable, then any valuation will make at least two clauses false. Again we keep the notation ϕ for the transformed formula and assume it contains n clauses and m variables.

Assuming that both transformations have been applied, we then build the turn-based game depicted on Fig. 3, which roughly contains two copies of the game we used in the NP-hardness proof.

The instance (ϕ, ϕ') of SAT-UNSAT is then positive iff any minimal coalition excluding q_0 has size exactly $n+2m+n'+2m'+1$. Indeed:

- if the instance is positive, then there exists a valuation of variables x_i satisfying ϕ. Applying the same construction as in the NP-hardness proof, a satisfying valuation yields a coalition of $n+2m$ players that has a winning strategy in the left part of the board. For ϕ', as for the other formula, it suffices to add all the "clause" and "question-mark" players to the coalition, as well as all "positive" variable players, plus $\neg z'$. This coalition is winning in the right part of the board thanks to the *almost-satisfiability* of ϕ'. This yields a winning coalition having $n+2m+n'+2m'+1$ players. By using the same arguments as in the NP-hardness proof, we can prove that no smaller coalition can win.
- conversely, assume that our SAT-UNSAT instance is negative. There may be several cases: (i) if both ϕ and ϕ' are satisfiable, then there is a winning coalition of size $n+2m+n'+2m'$; (ii) if both ϕ and ϕ' are unsatisfiable, then the coalition needs at least $n+2m+1$ players for winning in the left part of the board, and $n'+2m'+1$ players for winning on the right part. Thus at least $n+2m+n'+2m'+2$ players are needed to win the game; (iii) last, if ϕ is not satisfiable and ϕ' is, then exactly $n'+2m'$ players are necessary and sufficient to win on the right part of the board. But thanks to our transformation on ϕ, $n+2m+1$ players are not sufficient for winning on the left part, because we duplicated the original formula. Thus, again in that case, at least $n+2m+n'+2m'+2$ players are needed to win the game.

The extension to the unconstrained and co-constrained versions of the problem are similar to the NP-hardness proof, and we omit them. □

In the end, we have:

Theorem 10. *In explicit CGSs and for reachability objectives, the ((co)-constrained) coalition size problem is* NP*-complete, while the optimal coalition size problem is* DP*-complete.*

4. Related problems

In this section, we establish the precise complexity of several related problems. The first part is devoted to the study of *symbolic* CGSs, where the transition table is encoded symbolically. We then look at *function problems* directly related to our original problem: the aim is then to effectively compute a coalition (if any). Finally, we extend our results to different kinds of winning objectives, in particular quantitative objectives on durational CGSs.

4.1. Optimal coalitions in symbolic CGSs

As mentioned in Remark 2, symbolic CGSs are (assumed to be) more succinct than explicit ones. In particular, deciding whether a coalition has a winning strategy (for reachability, safety or Büchi objectives) can only be decided in Σ_2^{P} for this class of CGSs [14]. As a consequence, our algorithm for deciding the optimal coalition problem is now in Σ_2^{P} (i.e., the class $\mathsf{NP}^{\mathsf{NP}}$, see Appendix A for details): it consists in non-deterministically guessing a coalition of the given size together with a (memoryless) strategy for that coalition, and then check if the strategy is winning. The NP oracle is used for deciding which of the formulas defining the transition table are satisfiable. This algorithm is optimal, as it is possible to encode the Σ_2^{P}-complete problem

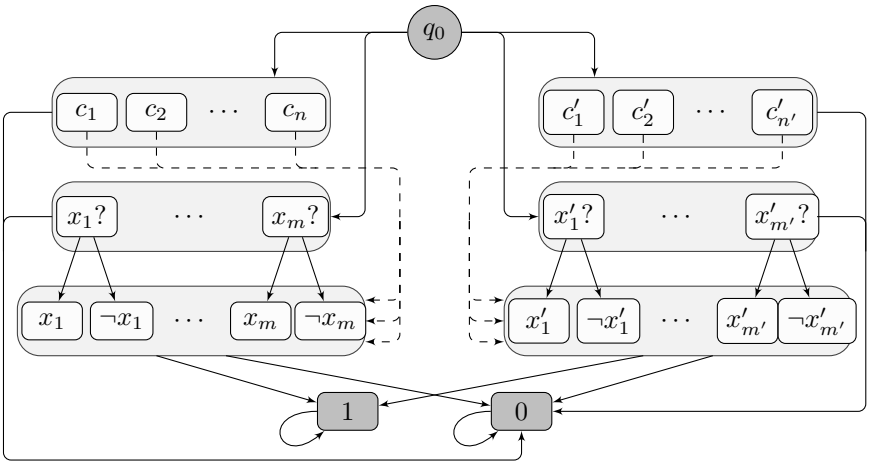

Figure 3. Global schema of the turn-based game used for the reduction to SAT-UNSAT.

QSAT_2 (see Appendix A) into the coalition size problem on symbolic CGSs. Thus, we have:

Theorem 11. *The ((co-)constrained) coalition size problem for reachability objectives is Σ_2^{P}-complete in symbolic CGSs.*

For the exact optimal coalition problem, we define a class similar to DP for Σ_2^{P} (see [25] and Appendix A):

$$\mathsf{D}_2^{\mathsf{P}} = \{L_1 \cap L_2 \mid L_1 \in \Sigma_2^{\mathsf{P}}, L_2 \in \mathsf{co}\Sigma_2^{\mathsf{P}}\}.$$

That our problem belongs to this class is rather obvious, with an argument similar to the one for explicit CGSs. It turns out to be complete for this class, as the encoding of SAT-UNSAT carried out in the proof of Theorem 9 can be adapted to encode the $\mathsf{D}_2^{\mathsf{P}}$-complete problem $\mathsf{QSAT\text{-}UNQSAT}_2$ (see Appendix A).

Theorem 12. *In symbolic CGSs, the optimal coalition size problem for reachability objectives $\mathsf{D}_2^{\mathsf{P}}$-complete.*

4.2. Computing optimal coalitions

Our problem of *deciding the existence* of a small coalition is obviously associated with the problem of *effectively computing* the size of an optimal coalition, or an optimal coalition itself. Those problems are called *function problems*, as opposed to classical *decision problems* [18]. Some of the associated complexity classes are described in Appendix A.

From our reduction of 3SAT (proof of Theorem 8), and since the function problem associated with 3SAT is FNP-complete, we immediately get hardness in FNP for the problem of computing a coalition of size less than k. This problem is obviously in FNP, as it suffices to find a witnessing coalition, which can be checked in polynomial time.

Proposition 13. *Finding a ((co)-constrained) winning coalition of size less than k for reachability objectives is FNP-complete on explicit CGSs.*

An algorithm for computing the size of an optimal winning coalition is by a binary search, using our NP-algorithm for the optimal coalition problem. This is achieved by a logarithmic number of calls to this NP-algorithm, yielding a algorithm in $\mathsf{FP}^{\mathsf{NP}[\log(n)]}$. Hardness of our problem in this class can be proved by encoding MAXSAT SIZE, where the aim is to compute the maximal number of clauses that are satisfiable at the same time in a 3SAT instance. This is achieved by slightly adapting our original reduction (and adding extra players) so that k is the maximal number of satisfiable clauses iff the optimal coalition size is $M - k$ for some integer M.

Theorem 14. *Computing the size of an optimal winning coalition for reachability objectives in explicit CGSs is $\mathsf{FP}^{\mathsf{NP}[\log(n)]}$-hard.*

The problem of effectively computing an optimal coalition can then be proved in $\mathsf{FP}^{\mathsf{NP}}$ with the following algorithm (adapted from [18, Example 10.4]): first compute the size S of the optimal winning condition, using the above binary search algorithm. Then apply the following procedure successively for each player p:

- replace each state s that p controls with a module similar to the one depicted on Fig. 2, with height and width $S + 1$. The incoming edges of s are plugged at the upper state of the module, while its outgoing edges are plugged on each of the bottom states. The upper state is controlled by p, and the each newly added state is controlled by a fresh player.
- compute the size of the optimal winning coalition in that game, using the NP algorithm.
- if the optimal size is unchanged, then there exists an optimal winning coalition not involving p, and we can

continue the algorithm with the new game; otherwise, player p is needed in any optimal winning coalition, and we continue applying the algorithm with the previous game (with states s instead of the triangular modules).

This algorithm runs in polynomial time with an NP oracle, and is thus in $\mathsf{FP}^{\mathsf{NP}}$. We were not able to close the gap between the $\mathsf{FP}^{\mathsf{NP}[\log(n)]}$ lower bound and the $\mathsf{FP}^{\mathsf{NP}}$ upper bound. To the best of our knowledge, the same situation occurs *e.g.* for MAX-CLIQUE, where the aim is to find a maximal clique in a graph [11].

Remark. *The algorithm above could be used for solving "ATL queries", following the ideas of [3]: for example, solving formula* $\mathsf{AG}(\langle\!\langle A \cup ? \rangle\!\rangle \phi)$ *consists in finding the smallest coalitions that, together with* A*, can enforce* ϕ *from any reachable state. This can be computed using ideas similar to the ones developed above.*

4.3. Priced agents

As mentioned in the introduction, an interesting extension of our original problem is the case where controlling an agent has a *price*: the input is extended with an array assigning this positive (binary-encoded) price to each agent, with the aim of finding the cheapest winning coalitions.

The complexities of the corresponding decision problems are the same as in the original case: the hardness proofs still apply, and the corresponding algorithms are easily adapted to handle the price. Regarding the function problems, the situation is different:

Theorem 15. *Computing (the price of) a cheapest winning coalition for reachability objectives in explicit CGSs with weighted agents is* $\mathsf{FP}^{\mathsf{NP}}$*-complete.*

The optimal price can be computed by binary search, and is proved optimal by encoding MAXSAT WEIGHT. This immediately yields hardness for the problem of computing an optimal coalition. Such an optimal coalition can be computed using the same technique as depicted in Appendix A for the *traveling salesman problem*.

4.4. Beyond reachability objectives

Since the game structures used in our proofs are mostly acyclic (the only cycles occur on states 0 and 1, and are only present for the sake of coherence with our definition of a run), our results obviously extend to several kinds of objectives, such as safety objectives and Büchi objectives.

More complex objectives can also be considered, for instance LTL objectives. The standard way of deciding whether a coalition has a winning strategy for such objectives is through non-deterministic Büchi and deterministic Rabin automata. This algorithm would then run in deterministic doubly-exponential time (but only singly-exponential in the number of agents). The problem of computing the (exact) optimal coalition size is then also in deterministic doubly-exponential time, since there are "only" exponentially many coalitions to test.

Hardness in 2EXPTIME can be proved as follows: consider a 2-player CGS, and a formula $\langle\!\langle A \rangle\!\rangle \phi$, where ϕ is an LTL formula. Deciding whether this formula holds in a given location q_0 of the CGS is 2EXPTIME-complete [1]. Now, if we add an extra move to player A from q_0 leading to a sink state (assumed to immediately make ϕ false), then clearly A must belong to any winning coalition, so that there is a winning coalition of size at most 1 iff player A has a strategy for the objective ϕ.

4.5. Quantitative objectives

The model of CGSs can be enriched by a duration function to obtain the model of DCGSs introduced in [13]. This model associates a duration with each transition of the CGS, allowing to model, for instance, a *"simple"* notion of time, or a notion of *cost*, and to verify quantitative properties on these systems.

Definition 16 ([13])**.** *A* Durational Concurrent Game Structure *(*DCGS *for short)* $\mathcal{C}$ *is a structure* $(\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathsf{Mv}, \mathsf{Edg}, \mathcal{D})$ *where* $(\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathsf{Mv}, \mathsf{Edg})$ *is a CGS, and* $\mathcal{D} \colon \mathsf{Edg} \to \mathbb{N}$ *associates a nonnegative duration with each transition.*

Given a finite execution $\rho_f = q_0 q_1 \cdots q_n$, we can naturally associate a duration $\mathcal{D}(\rho_f)$ with ρ_f. When interested in a reachability objective (of a set F), we denote by $\mathcal{D}(\rho)$ the duration of the shortest finite prefix of ρ ending in F, if such a finite prefix does not exist, we say that $\mathcal{D}(\rho)$ is infinite.

In the framework of DCGSs, quantitative versions of the reachability problem naturally arise. For instance, given a coalition A and a nonnegative integer d, one can ask whether coalition A can ensure its reachability objective within duration at most d. More formally, we ask whether there exists a winning strategy $(f_i)_{A_i \in A}$ (for the reachability objective) such that for each $\rho \in \mathrm{Out}(q_0, (f_i)_{A_i \in A})$, we have that $\mathcal{D}(\rho) \leq k$. In [13], an algorithm is proposed for this kind of problem with positive costs. It runs in PTIME if no equality is involved, and in EXPTIME in the general case.

In the spirit of this paper, various *optimal coalition problems on DCGS* could be considered, where we would ask to minimize both the size of the winning coalition and the duration needed to achieve the objective. We only define and examine two variants of these problems.

Definition 17. *The problem of* coalition size with bounded duration *is defined as follows: given a DCGS* $\mathcal{C}$*, an initial location* $q_0 \in \mathsf{Loc}$*, a set of states* $F \subseteq \mathsf{Loc}$ *to be reached, a*

maximal size $k \in \mathbb{N}$, a maximal duration $d \in \mathbb{N}$; determine if there exists a coalition of size at most k having a winning strategy from q_0 for reaching F with duration at most d.

Given a coalition A, deciding if A has a winning strategy to reach F within a maximal duration d can be done in PTIME [13]. As previously, this leads to an NP algorithm for the coalition size problem with bounded duration.

On the other hand, the coalition size problem with bounded duration is clearly NP-hard, since it contains the coalition size problem as a special case (by letting all durations be zero). This straightforward extends to:

Theorem 18. *In DCGSs and for reachability objectives, the ((co)-constrained) coalition size problem with bounded duration is* NP*-complete, while the optimal coalition size problem with bounded duration is* DP*-complete.*

Last, we turn to the extension to exact durations:

Definition 19. *The problem of* optimal coalition with exact duration *is defined as follows: given a DCGS $\mathcal{C}$, an initial location $q_0 \in \mathsf{Loc}$, a set of states $F \subseteq \mathsf{Loc}$ to be reached, a maximal size $k \in \mathbb{N}$, a maximal duration $d \in \mathbb{N}$; determine if there exists a coalition of size at most k having a winning strategy from q_0 for reaching F with duration exactly d.*

Given a DCGS $\mathcal{C}$ and a coalition A, deciding if A can reach F with duration exactly d can be done in exponential time (precisely in time $O(|\mathcal{C}|^2 d)$, with d encoded in binary [13]). Executing this algorithm once for each coalition of size k leads to an EXPTIME algorithm for the coalition size problem with exact duration.

Given a two-player DCGS, the problem of deciding if the first player has a winning strategy to reach F with duration exactly d has been proved EXPTIME-complete in [13]. This problem can easily be encoded in the coalition size problem with exact duration. This leads to the following theorem:

Theorem 20. *The coalition size problem with exact duration is* EXPTIME*-complete.*

Since we end up in a deterministic class, deciding whether k is the optimal size remains EXPTIME-complete.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, Sept. 2002.

[2] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. of the AMS*, 138:295–311, Apr. 1969.

[3] W. Chan. Temporal-logic queries. In *Proc. 12th Intl Conf. Computer Aided Verification (CAV'00)*, LNCS 1855, p. 450–463. Springer, July 2000.

[4] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *Intl Journal Software Tools for Technology Transfer*, 2(4):410–425, Mar. 2000.

[5] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, LNCS 131, p. 52–71. Springer, 1981.

[6] V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artif. Intell.*, 170(6-7):607–619, 2006.

[7] V. D. Dang, R. K. Dash, A. Rogers, and N. R. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proc. 21st Nat. Conf. Artificial Intelligence (AAAI'06) and 18th Innovative Applications of Artificial Intelligence Conf. (IAAI'06)*. AAAI Press, 2006.

[8] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proc. 8th Conf. Theoretical Aspects of Rationality and Knowledge (TARK'01)*, p. 259–272. Morgan Kaufmann Publishers, July 2001.

[9] G. J. Holzmann. The model checker SPIN. *IEEE Trans. Software Engineering*, 23(5):279–295, May 1997.

[10] S. Hornus and Ph. Schnoebelen. On solving temporal logic queries. In *Proc. 9th Intl Conf. Algebraic Methodology and Software Technology (AMAST'02)*, LNCS 2422, p. 163–177. Springer, Sept. 2002.

[11] M. W. Krentel. *The Complexity of Optimization Problems*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, USA, 1987.

[12] S. A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[13] F. Laroussinie, N. Markey, and G. Oreiby. Model checking timed ATL for durational concurrent game structures. In *Proc. 4th Intl Conf. Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, LNCS 4202, p. 245–259. Springer, Sept. 2006.

[14] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. In *Proc. 10th Intl Conf. Foundations of Software Science and Computation Structures (FoSSaCS'07)*, LNCS 4423, p. 243–257. Springer, Mar. 2007.

[15] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Mathematical Biophysics*, 5:115–133, 1943.

[16] K. L. McMillan. *Symbolic Model Checking — An Approach to the State Explosion Problem*. PhD thesis, CMU, 1993.

[17] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.

[18] Ch. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[19] Ch. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing (STOC'82)*, p. 255–260. ACM Press, May 1982.

[20] A. Pnueli. The temporal logic of programs. In *Proc. 18th Ann. Symp. Foundations of Computer Science (FOCS'77)*, p. 46–57. IEEE Comp. Soc. Press, 1977.

[21] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglini and U. Montanari, editors, *Proc. 5th Intl Symp. Programming (SOP'82)*, LNCS 137, p. 337–351. Springer, Apr. 1982.

[22] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, Jan. 1989.
[23] T. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, 1997.
[24] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
[25] M. Wooldridge and P. E. Dunne. On the computational complexity of qualitative coalition games. *Artificial Intelligence*, 158(1):27–73, Sept. 2004.

A Some notes about complexity classes

We briefly define some unusual complexity classes that we use troughout the paper, together with some related problems. We assume the reader is familiar with PTIME and NP. The interested reader will find much more details in [18].

A.1 Complexity of decision problems

The polynomial-time hierarchy. The polynomial hierarchy, introduced in [24], is an infinite hierarchy of classes that lie between NP and PSPACE. These classes are defined in terms of *oracle Turing machines*: given a complexity class $\mathcal{C}$, a Turing machine with oracle in $\mathcal{C}$ is a Turing machine equipped with a special device (the *oracle*), which can answer in constant time to a problem in $\mathcal{C}$. We refer to [18] for a more formal definition.

Applying restrictions on the computational power of those machines, we get new complexity classes: for instance, $\mathsf{PTIME}^{\mathsf{NP}}$ is the class of problems that are solvable in deterministic polynomial time on a Turing machine equipped with an NP-oracle. The *polynomial hierarchy* is the hierarchy of complexity classes based on this construction:

$$\Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \Delta_0^{\mathsf{P}} = \mathsf{PTIME} \qquad \Delta_{i+1}^{\mathsf{P}} = \mathsf{PTIME}^{\Sigma_i^{\mathsf{P}}}$$
$$\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}} \qquad \Pi_{i+1}^{\mathsf{P}} = \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}$$

Formally, the polynomial hierarchy, denoted by PH, is the union of all those classes. It is easy to see that, at any finite level i, the three classes Δ_i^{P}, Σ_i^{P} and Π_i^{P} are contained in PSPACE. It is not known whether the hierarchy is strict or collapses at some level, nor whether $\mathsf{PH} \subseteq \mathsf{PSPACE}$.

At any finite level, those classes all have complete problems. For Σ_i^{P}, a classical example is the problem QSAT_i, defined as follows: given i finite sets $(X_j)_{j \leq i}$ of variables, and a boolean formula Φ over $\bigcup_{j \leq i} X_j$, is it true that $\exists X_1 . \forall X_2 \ldots Q_i X_i \Phi$, where quantifiers alternate and Q_i is either $\exists$ or $\forall$, depending on the parity of i. It is straightforward to prove that QSAT_i is in Σ_i^{P}. The hardness proof is more involved, see [18].

It is possible to further refine those classes by imposing a bound on the number of calls to the oracle. For instance, $\mathsf{PTIME}^{\mathsf{NP}[\log n]}$ denotes the subclass of Δ_2^{P} in which only a logarithmic number of call to the oracle is allowed.

Difference classes. Difference classes are classes of languages defined as the difference of two classes. They were introduced in [19]. The best-known difference class is $\mathsf{DP} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP},\ L_2 \in \mathsf{coNP}\}$. Note that DP is not $\mathsf{NP} \cap \mathsf{coNP}$: the former contains NP (and coNP), while the latter is expected to be a strict subclass of NP. Note also that DP is a subclass of Δ_2^{P}.

There exists several known problems that are complete for DP: SAT-UNSAT is the set of couples (ϕ, ϕ') of boolean formulas s.t. ϕ is satisfiable and ϕ' is not. This problem is easily seen to be solvable in DP, and can be proved to be DP-complete.

Of course, many other difference classes can be used, in particular $\mathsf{D}_2^{\mathsf{P}} = \{L_1 \cap L_2 \mid L_1 \in \Sigma_2^{\mathsf{P}},\ L_2 \in \Pi_2^{\mathsf{P}}\}$. An example of a complete problem is $\mathsf{QSAT\text{-}UNQSAT}_2$, defined as the set of couples $(\exists X . \forall Y . \phi(X, Y), \exists X' . \forall Y' . \phi'(X', Y'))$, the first one being a positive instance of QSAT_2 and the second one being a negative instance.

A.2 Complexity of function problems

Contrary to decision problems where a yes/no answer is sufficient, function problems consists in computing a solution to a problem. An obvious example is FSAT, which is given a boolean formula and must compute a satisfying assignment (or return "*no*" if none exists). This defines the class FNP of function problems associated to problems in NP. The class FP is the subclass associated to problems in PTIME. As in the case of decision problems, a notion of "reduction" between problems can be defined, and FNP contains complete problems (*e.g.*, FSAT).

Oracles can also be used in the setting of function problems: the class $\mathsf{FP}^{\mathsf{NP}}$ contains function problems that can be computed in polynomial time with the use of an NP oracle. The famous *traveling salesman problem*, in which the actual optimal tour must be computed, is complete for $\mathsf{FP}^{\mathsf{NP}}$: the algorithm consists in first computing the optimal length by binary search, and then computing an optimal tour by testing if the same length can be achieved after incrementing the intercity distance one after the other: in case it can, then that particular edge can be avoided in the optimal tour. This yields a $\mathsf{FP}^{\mathsf{NP}}$ algorithm for TSP, which can be proved complete for this class.

Finally, we can again restrict the number of calls to the oracle, and define the class $\mathsf{FP}^{\mathsf{NP}[\log n]}$ with the expected definition. An example of a complete problem for this class is MAXSAT SIZE, in which, given a set of disjunctive clauses, we want to compute the maximum number of clauses that can be satisfied at the same time (algorithm by binary search). If clauses are assigned a weight and we want to compute the maximal total weight of satisfied clauses, we get MAXSAT WEIGHT, which is $\mathsf{FP}^{\mathsf{NP}}$-complete.