Factorization of Dynamic Games over Spatio-Temporal Resources

Author(s):
Zanardi, Alessandro; Bolognani, Saverio; Censi, Andrea; Dörfler, Florian; Frazzoli, Emilio

Publication date:
2022

Permanent link:
https://doi.org/https://doi.org/10.3929/ethz-b-000560629

Originally published in:
https://doi.org/10.1109/IROS47612.2022.9981416

Funding acknowledgement:
- NCCR Automation (phase I) ()

Factorization of Dynamic Games over Spatio-Temporal Resources

Alessandro Zanardi[1*], Saverio Bolognani[2], Andrea Censi[1], Florian Dörfler[2], Emilio Frazzoli[1]

***Abstract*— Dynamic games feature a state-space complexity that scales superlinearly with the number of players. This makes this class of games often intractable even for a handful of players. We introduce the *factorization* process of dynamic games as a transformation leveraging the independence of players at equilibrium to build a leaner game graph. When applicable, it yields fewer nodes, fewer players per game node, hence much faster solutions. While for the general case checking for independence of players requires to solve the game itself, we observe that for dynamic games in the robotic domain there exist exact heuristics based on the spatio-temporal occupancy of the individual players. We validate our findings in realistic autonomous driving scenarios showing that already for a 4-player intersection we have a reduction of game nodes and solving time close to 99%.**

I. Introduction

As robotics becomes ubiquitous, embodied intelligence cannot neglect to cope with multi-agent environments. Despite many sub-fields of multi-agent decision making (e.g., game theory, distributed optimization, multi-agent learning [1]) having received a lot of attention in the last decade, a persistent challenge to overcome is the superlinear growth in complexity in the number of competitive players [2]. In addition, perhaps because of the human tendency to subconsciously underestimate the exponential-growth bias [3], many of the available approaches work for 2-3 players, but become brittle for a handful, and break with more.

There are in general two ways to overcome the scalability issue of games with several agents: Either to settle for approximate solutions, or to exploit a particular structure of the problem at hand. In this work, we focus on the latter, considering the robotic motion planning problem, formulated as a non-cooperative dynamic game, as the guiding application. Many of the works in this field look at fast local solvers for relatively short prediction horizons, which exhibit good interactive behaviors when implemented in a receding horizon approach [4]–[9]. The hardness of these problems is inborn in the superlinear (almost exponential) explosion of number of states and joint actions for the game [10], [11]. The deeper the game graph and the more the players, the worse for the problem.

Identifying that the problem at hand has an inherent constrained interaction pattern among the players, such as in social media, transportation, or power grids, has already proven to be successful. A seminal answer in this regard was given by Graphical Games [12], which reformulate the interconnection of players over a graph to a game where each player's payoff depends only on its neighbors' actions.

As observed in [10] though, for many other real-world non-cooperative games the interaction structure among the players is not fixed or known a priori. This is the case also for the application at hand. Autonomous Vehicles (AVs)' strategic decisions on the roads clearly depend only on few other relevant vehicles. Yet these are hard to identify and constantly changing, even within the planning horizon.

In this work, we consider the motion planning problem formulated as a non-cooperative dynamic game and aim for a tool that dynamically *factorizes* the stage subgames into smaller ones while maintaining a notion of strategic equivalence to the original game. This allows one, for instance, to decouple the solution of a n-player subgame from certain states onwards into the composition of solutions of smaller subgames; n 1-player games in the best case. We showcase that this approach has a tremendous impact in taming the superlinear complexity of the problem at hand in realistic scenarios.

A. Related Work

The idea of exploiting strategic dependency patterns among the players to transform the game into a simpler form has already been a key pillar in Graphical Games [12], Action Graph Games [13], and Multi-agent Influence Diagrams [2]. When available, the influence structure allows more compact representations that speed up the solution process. The influence diagrams are often represented with the language of Bayesian Networks to represent conditional independence given a certain decision for the other players [14]–[17]. Similar formulations employ the same structure conditioning on events, as in Chain Event Graphs [18], [19].

We believe that to be applicable for robotic motion planning, these ideas need to be revisited. In particular, the game of interest are multistage, played in state space, with no prior information on the dependency structure that can also dynamically change as the stages advance. Very little of these key ingredients are found in the mentioned works. To the best of our knowledge, the idea of finding decompositions of the game (factorizations) that are not known a priori to support a "divide and conquer" approach to speed up the finding of Nash equilibria is present only in [10], where *factors* in the representation correspond to connected components in a previous graphical representation for games. To this end, we consider a factorization process that naturally applies to multistage dynamic games inferring the strategic independence for each state on-line while we build the game graph.

*This work was supported by the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40_180545.

[1]A. Zanardi, A. Censi, and E. Frazzoli are with the Institute for Dynamic Systems and Control, ETH Zürich, Switzerland {azanardi}@ethz.ch.

[2]S. Bolognani and F. Dörfler are with the Automatic Control Lab, ETH Zürich, Switzerland.

B. Statement of Contribution

The contribution of this work is threefold. First, we introduce the *factorization* of dynamic games as a systematic transformation that puts the original game in a leaner form, which is simpler to solve and still retrieves exact solutions. A factorized dynamic game can have in general several smaller subgames to solve, each with fewer players. Second, we observe that for dynamic games in the realm of multi-agent motion planning, there exist simple heuristics for factorization based on the spatio-temporal resources required by the players. We introduce two algorithms where these resources are computed by solving first the individual Dynamic Programming (DP) problems of the players. Finally, we instantiate these factorization algorithms in the domain of AVs.

We take realistic state-of-the-art scenarios from CommonRoad [20] and quantitatively show that, as the complexity of the test cases increases with the number of players, building and solving the factorized dynamic game has a tremendous impact on computational performance. For a complex multi-lane intersection, already with 4 players the factorized game retrieves the solution of the original game solving 99% fewer subgames, each containing fewer players. This is reflected also in total solving times which have similar order of magnitude of relative improvements, showing that the factorization overhead is completely negligible.

II. Preliminaries: Dynamic Games

The building block of this work are discrete dynamic games played in state space with state feedback structure (often called Markovian) [21], [22]. Focusing on embodied agents, we consider only games whose state can be decoupled into the individual states of the players. Hence we consider dynamic games defined by a finite set of players $\mathcal{A}$; each player $i \in \mathcal{A}$ has a corresponding state space X_i and an action set U_i.

Each player is subject to a cost structure modeled by an incremental stage cost as a mapping $g_i : X \times U \to O_i$ and a terminal cost $s_i : X \to O_i$, where O_i denotes the outcome set for the i-th player. We assume it to be a totally ordered set (e.g., real numbers, lexicographic preference,...) with a monoidal structure, which allows stage-additive costs while respecting the Bellman optimality principle [23]. Note that real-valued outcomes clearly satisfy this assumption.

To ensure themself a better outcome each player aims to find a strategy, i.e. a state feedback policy, possibly stochastic, of the form $\gamma_i : X \to \Delta(U_i)$. The multi-stage nature is captured by the game dynamics that we assume to be separable with respect to the individual players up to collisions. We denote the dynamics of a player as $\theta_i : X_i \times U_i \to X_i$. Given an initial condition x^0, each player aims to get their best possible cumulative stochastic outcome by an appropriate selection of strategy γ_i. The cumulative outcome over a finite horizon T_i is

$$\begin{aligned} o_i(\gamma|x^0) &= \mathbb{E}_\gamma \sum_{k=0}^{T_i-1} g_i(x(k), \gamma(x(k))) + s_i(x(T_i)), \\ \text{s.t.} \quad x(k+1) &= \theta(x(k), u(k)), \quad x(0) = x^0. \end{aligned} \tag{1}$$

Notation: We use subindices to denote specific quantities relative to a single or a subset of players. Symbols with no index are used to denote the joint quantities of all the players.

A. Game Solutions

Solving a dynamic game involves finding an equilibrium strategy such that no player is better off by unilaterally changing its strategy, the so-called Nash Equilibrium (NE).

Definition 1 (Feedback NE)**.** *A strategy profile* $\boldsymbol{\gamma}^* = [\gamma_i^*, \gamma_{-i}^*]$ *is a* feedback NE *if and only if:*

$$o_i(\boldsymbol{\gamma}^*|x) \preceq o_i(\gamma_i, \gamma_{-i}^*|x) \quad \forall i \in \mathcal{A} \text{ and } \forall \gamma_i \in \Gamma_i, \tag{2}$$

for any *reachable point* x*.*

Commonly in the literature, such a strategy (Def. 1) is also called subgame perfect [21]. When finding such strategies becomes impractical and computationally cumbersome, one might want to settle for "weaker" solution concepts. Most notably, open loop NE that yields the optimal sequence of actions only for a specific initial state or for a limited look-ahead horizon (e.g., MPC-like solvers). The interested reader is referred to [21], [24] for further details on different possible solution concepts in multi-stage games. Independently of the solution concept of choice, we denote by $\textsc{Sol}(\mathbf{G}(x))$ the set of strategies satisfying the equilibrium criterion for a game rooted in x. These can in general be computed by performing backward induction on the game graph [21].

For an easier understanding of the remaining, recall that a natural representation of the game dynamics is a graph on the joint state space. Thus the term "game graph". Each node is uniquely identified by the joint state of all the players and the edges represent the possible transitions. We will see that a slightly more general notion of "game node" and transitions are key to the factorization process.

III. Factorization of Dynamic Games

We introduce the *factorization* of a dynamic game as a process that transforms the original game into a leaner and simpler structure to solve, but still retrieving the solution of the original game. The key insight is observing that some players, from certain states onwards, can result decoupled. Importantly here, "decoupled" does not necessarily mean that they have no strategic influence on each other, rather that the solution of the remaining game can be found composing the solutions of two separate subgames. This intuition is further represented in the example of Fig. 1.

The reformulation requires only a subtle, yet significant, change. While in the textbook formalization of dynamic games, the nodes of the game are identified with the joint states of *all* the players, for a factorized game, we allow game nodes to contain only a subset of the players (and their respective state). For a certain game node, the strategies, the incremental costs, and the dynamics of the players will be dependent only on the other players present in the same node. Moreover, the transitions (dynamics of the game) happen among game nodes, allowing also players to transition to different game nodes. For instance, two players currently in the *same* game node can transition to *different* future

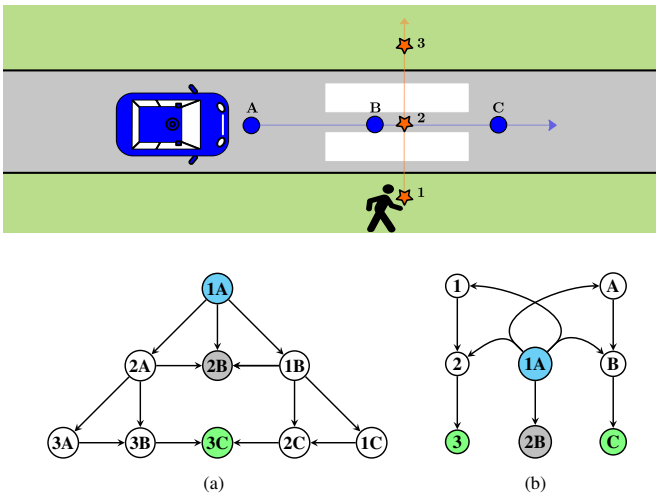

Fig. 1. A toy dynamic game where two players aim to reach the end as quickly as possible with the possibility, at every time instant, of advancing by one step or waiting. The resulting game graph *without* factorization is visualized in (a). Starting from the joint state $\{1A\}$ only 2 optimal paths exist to reach the desired end state $\{3C\}$ without collisions. Note that in order to find these paths one has to solve 7 game nodes with 2 players ($3C$, $2B$ are terminal nodes). Self-loops and "early exits" of players that would lead to additional individual game nodes are omitted for simplicity. In (b), on the right, we display the factorized counterpart. Interestingly, the two (open loop) equilibria for the game can be computed solving only 1 2-player game. In fact, after the first decision is taken by the agents, and excluding the terminal node $2B$, the decisions processes by the two agents are strategically decoupled.

game nodes. Once in two different nodes, they will result decoupled in terms of outcomes, strategies, and transition for the remaining of the game.

Having fewer and smaller game nodes to solve has paramount importance to curb the computational complexity of solving dynamic games. The joint action space increases exponentially with the number of players [25], m^n for n players each with m actions. As a consequence, not only equilibria would be computationally harder to find, but also the number of states of the game grows exponentially in the number of players rather than linearly. Factorization tackles this issue by decoupling players as much as possible while still being able to retrieve the solution of the original game. We now formalize this intuition and provide conditions for factorization to be well-posed.

Factorized games and equivalence of solutions

We consider a game node for a factorized dynamic game to be an independent subgame containing (possibly) only a subset of all the players and their states. More formally:

$$\mathbf{G}(x_{\mathcal{A}^{\mathbf{G}}}) = \langle \mathcal{A}^{\mathbf{G}}, \{x_i\}, \{\Gamma_i\}, \{O_i\}, \{\preceq_i\};\ i \in \mathcal{A}^{\mathbf{G}} \subseteq \mathcal{A} \rangle .$$

In words, a subset of the players, their *current* state, the available strategies, the stage outcomes, and the preference relation of the players on the outcomes (e.g., smaller is preferred). Note that this is different from a generic dynamic game definition, mostly because a game node is bound to the state of only a subset of all the players. The actions, strategies, and outcomes are again considered only with respect to the players in the game node, regardless of the state of the others. Accordingly, the stage costs (g, s), and the strategies γ shall be considered as mappings from a game node (and not the joint state of everyone) to their respective co-domain.

The dynamics has to change accordingly. Consider $\mathcal{G}$ to be the set of all game nodes, the dynamics for the factorized game is a function $\theta^{\mathsf{F}} : \mathcal{G} \to pow(\mathcal{G})$; where pow denotes the powerset. In contrast to the standard dynamics of a dynamic game, the factorized one allows players to transition to different successive game nodes. For instance, consider a game node with 2 players. The transitions $\{x_1, x_2\} \mapsto \{x_1^+, x_2^+\}$ and $\{x_1, x_2\} \mapsto \{\{x_1^+\}, \{x_2^+\}\}$ shall be considered different. In the latter the two players transition to separate game nodes, hence decoupling them for the remaining of the game. Note the players can only get decoupled as the game progresses. Solution concepts stay the same, even though for each game node the equilibrium is reached among the players present in the node.

Clearly not all possible graphs of game nodes are valid. We are ultimately interested in the ones that factorize the original game, and ultimately permit to retrieve one of its solutions. Some coherency conditions are due.

Definition 2 (Valid Factorization)**.** *We say that a factorization of a dynamic game is* valid *if and only if*

1) *Given any joint state x of the original game there exist a unique set of game nodes $\mathcal{G}(x) = \{\mathbf{G}', \mathbf{G}'', \ldots\}$ such that $x = [x^{\mathbf{G}'}, x^{\mathbf{G}''} \ldots]$. I.e., there exists a unique set of game nodes that can "reconstruct" a non-factorized state.*
2) *The factorized dynamics induces a direct graph on the game nodes such that for every transition $\mathbf{G} \mapsto \mathcal{G}^+$ it needs to hold $\mathcal{A}^{\mathbf{G}} \cap \mathcal{A}^{\mathbf{G}'} = \emptyset \ \ \forall \mathbf{G} \neq \mathbf{G}' \in \mathcal{G}^+$. I.e., players can only get decoupled as the game evolves in time, and a player cannot be present in different game nodes at the same stage.*

In the remaining we assume all the considered factorizations to be valid. We are left with comparing a solution of the factorized game with a solution of the original game.

Definition 3 (Equivalent strategies)**.** *Consider γ^{F} to be a strategy profile for a valid factorization of the game. An equivalent strategy profile γ for the original game can then be constructed by the product of factorized ones. Hence we say γ^{F} and γ are equivalent if and only if $\forall x \in X$:*

$$\gamma(x) \overset{1}{=} [\gamma^{\mathsf{F}}(x^{\mathbf{G}}), \ldots] \quad \textit{with } x = [x^{\mathbf{G}}, \ldots]^2,$$

up to reordering of the players.

Thus, we introduce the following definition, parametrized by the specific solution concept.

Definition 4 (Equivalence of Solution)**.** *Let $(\gamma^{\mathsf{F}})^* \in \textsc{Sol}(\mathbf{G}(x^0))$ be a solution of the factorized game. We say that $(\gamma^{\mathsf{F}})^*$ is a solution of the original game if and only if its equivalent strategy $(\gamma)^*$ (according to Def. 3) is a solution for the original game.*

[2]Note that this is well-posed since for a valid factorization there is a unique way of reconstructing the state of all players.

A. Influence among players

The goal of the factorization process is to find partitions of players whose solutions can be computed independently, yet their composition must yield a solution of the original game. This can be seen as a way of finding a decomposition of the original game into a set of (simpler) games whose composition is strategically equivalent to the original one.

This property holds if the players are independent at equilibrium, more formally:

Definition 5 (Independence of Subgames at Equilibrium)**.** *Let us consider a dynamic game rooted in x, with (at least) two disjoint sets of players $\mathcal{A}', \mathcal{A}'' \subseteq \mathcal{A}$. Then consider the following factorization into subgames $\mathbf{G}(x_{\mathcal{A}'})$, $\mathbf{G}(x_{\mathcal{A}''})$. We say that $\mathcal{A}'$ are independent of $\mathcal{A}''$ at equilibrium, denoted as $\mathcal{A}' \perp\!\!\!\perp^* \mathcal{A}''$, iff*

$$\hat{\gamma} \doteq [\gamma^*_{\mathcal{A}'}, \gamma^*_{\mathcal{A}''}] \quad \hat{\gamma} \in \textsc{Sol}(\mathbf{G}(x)), \tag{3}$$

$$\forall \gamma^*_{\mathcal{A}'}, \gamma^*_{\mathcal{A}''} \in \textsc{Sol}(\mathbf{G}(x_{\mathcal{A}'})) \times \textsc{Sol}(\mathbf{G}(x_{\mathcal{A}''})).$$

In words, (3) is saying that two players are independent at equilibrium if and only if every combination of solutions of the factorized game is a solution of the original game. We remark that independence at equilibrium is only a strategic equivalence. It does not imply that the outcomes of the factorized games and the original are the same. Only that the players would choose the same strategies.

Similarly, we can consider a stronger notion of independence, for players that cannot influence each others for the remaining of the game, not just at equilibrium.

Definition 6 (Strategic Independence of Subgames)**.** *Consider the same setup of Def. 5. And denote the best response set of the i-th player by $\mathcal{R}_i()$. We say that $\mathcal{A}'$ are strategically independent of $\mathcal{A}''$, denoted as $\mathcal{A}' \perp\!\!\!\perp \mathcal{A}''$, iff*

$$\mathcal{R}_{\mathcal{A}'}(\gamma_{\mathcal{A}''}), \mathcal{R}_{\mathcal{A}''}(\gamma_{\mathcal{A}'}) \text{ are constant}$$

$$\forall \gamma_{\mathcal{A}''} \in \Gamma_{\mathcal{A}''}, \forall \gamma_{\mathcal{A}'} \in \Gamma_{\mathcal{A}'}$$

In words, their respective best response sets are invariant with respect to any strategy of the others. We have the following immediate result, that will come handy when factorizing.

Proposition 1. *Strategic independence implies independence of players at equilibrium.*

Proof. Follows trivially by Def. 5 and Def. 6 since the equilibrium strategy belongs to the set of best responses. ■

B. Influence Graph and Phase Transitions

Given a state of the original game, the independence Defs. 5 and 6 induce an influence graph among the players. In this representation, the lack of an edge between connected components represents a possible factorization among the players. It is important to notice that this (in)dependence holds only from the given state onwards. Clearly two players that are dependent at the current state can become independent in future states.

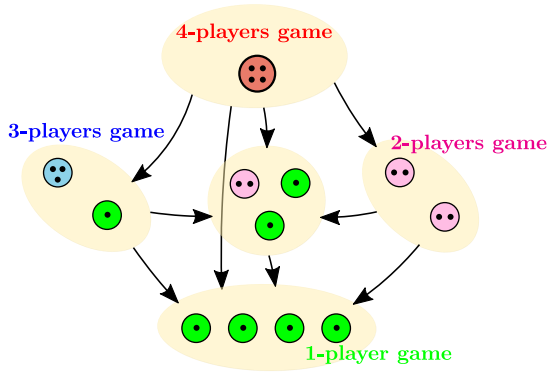

Fig. 2. The factorization phase diagram associated to a 4-player game. The arrows represent the only possible transitions that a valid factorization implies, from game nodes of higher cardinality to partitions of lower complexity. We highlight that for the *same* dynamic game, different reachable states can lead to different factorization phases for their respective subgames.

Definition 7 (Influence Graph)**.** *Let x be a joint state for players $\mathcal{A}$. We call the influence graph among the players in x the graph $\mathcal{I}(x) = \langle \mathcal{A}, \mathcal{A} \times \mathcal{A} \rangle$ where we remove all the edges (i, j) if Pi, Pj satisfy the independence relation Def. 6 for the subgames $\mathbf{G}(x_i), \mathbf{G}(x_j)$. Similarly, we call the influence graph at equilibrium $\mathcal{I}^*$ the same graph constructed removing the edges according to Def. 5.*

Definition 8 (Factorization phases)**.** *We call* factorization phases *the set of possible partitions (i.e., connected components of the influence graph) of the players up to relabeling of the players.*

Hence we can observe that a valid factorization admits only certain transitions among phases, graphically depicted in Fig. 2. Intuitively as the dynamics of the game evolves, the joint state of the players can be further factorized into smaller games. In regard to Fig. 2 we observe the following:

- In general for a factorized dynamic game only certain phase transitions can happen.
- The initial state could be already factorized. For instance, for a n-player dynamic we might never need to solve a n-player game node.
- For any given state of the original game there exist a unique corresponding phase.

C. Factorization algorithms as heuristics

The process of solving a dynamic game involves two steps: 1) building the game graph 2) solving each game node via backward induction. As a general rule, the smaller the number of nodes and the fewer the number of players, the better for computation. Thus, we would like to build the game graph "as factorized as possible".

When building the game graph, we expand each game node and get a candidate new state that we check for factorization. Can the new state be factorized into smaller subgames? Def. 5 tells us that we are looking for new subgames that can be solved independently to retrieve the solution of the non-factorized game. While Def. 5 defines what we would like to have, it has little applicability unless one already knows the solutions of the (sub)game and the candidate factorizations.

For this reason, we consider a factorization algorithm more similar to a heuristic function that must imply Def. 5 when applied[3]. Each algorithm for factorization might be more or less expensive to compute and yield a factorization that is more or less conservative. One extreme is given by the "identity", which never partitions the new states. On the other end, one could directly apply Def. 5 to each new candidate state. This yields the "slimmest" possible factorization, but at the price of a very high computational burden. We imagine that between these two extrema there is a whole Pareto-front to discover. In Alg. 1 we observe that FACTORIZE is indeed a function placeholder in the node expansion (line 12). In the following, we fill this placeholder for a specific class of games.

Algorithm 1 Build game graph with factorization.

```
1:  X ← {}                                          ▷ The processed states
2:  G ← {}                                          ▷ The game nodes
3:  for all x^F ∈ FACTORIZE(x^0) do
4:      BUILD_GAME_GRAPH(x^F)
5:  function BUILD_GAME_GRAPH(x)
6:      x̂ ←REMOVE_TERMINATED_PLAYERS(x)
7:      if x̂ ∈ G or x̂ == ∅ then return
8:      X.add(x̂)
9:      for all γ ∈ Γ(x̂) do
10:         G.add(GAME_NODE_FROM_STATE(x̂, γ))
11:         x^+ = θ(x̂, γ)                           ▷ Stage dynamics
12:         for all x^F ∈ FACTORIZE(x^+) do
13:             BUILD_GAME_GRAPH(x^F)               ▷ Recursive call
```

IV. Factorization Applied to Robotic Motion Planning

We now instantiate the concepts presented to dynamic games in the realm of robotic motion planning. It turns out that for this class of games we can compute inexpensive heuristics to factorize the game. The result is fewer nodes to solve and less required computation time.

While checking for independence of players can be cumbersome in the general case, many multi-agent motion planning problems can be formulated in a natural way as games where the coupling among the players happen only via a shared collision/proximity term in the cost function. Among many examples, see [4]–[9], [26]. The last decades of robotic developments taught us that a natural way of checking for collision and proximity involves associating an occupancy map to the state of the robot. Indeed, we think of robots as players competing for portions of the workspace–both in space and time. We observed that in these cases, we can factorize the game by checking for independence with a logical AND operation between reachable sets of resources.

A. Factorization using reachable resources

More formally, we consider a set of resources R as quantization of the workspace $\mathcal{W}$ and the time axis $\mathbb{T}$, that is $R \doteq \mathcal{W} \times \mathbb{T}$. We then associate a subset of the resources to each game node. Similarly to an incremental cost, we have

Definition 9 (Resources of a game node)**.** *Let* $\mathbf{G}$ *be a game node. A mapping of the type* $\mathsf{Res}(\mathbf{G}) : X^{\mathbf{G}} \times \Gamma^{\mathbf{G}} \to R$ *defines the resources associated to* $\mathbf{G}$*. The resources can be further conditioned on a specific strategy, denoted as* $\mathsf{Res}^{\gamma}(\mathbf{G})$.

For example, in the robotic context computing these resources corresponds to finding the occupancy map of the robots. Further, given the resources associated to each game node, one can compute recursively the reachable resources at the next stages. We denote the set of future reachable resources as a map $\mathsf{futureRes}(\mathbf{G}) : \mathbb{T} \times X^{\mathbf{G}} \times \Gamma^{\mathbf{G}} \to R$. When conditioned on a fixed strategy γ we write $\mathsf{futureRes}^{\gamma}(\mathbf{G})$. Note that, given a state and a future time index, the future resources correspond to the forward reachable set of a player at the given time [27].

Our main contribution with respect to this class of dynamic games for embodied agents is the following factorization procedure. Instead of building the game graph and solving it, we do:

1) Solve the DP problems of each of the single players;
2) Compute the reachable sets of resources associated to the individual players game (the DPs);
3) Build the factorized game graph using the intersection of future resources as a factorization heuristic for independence of players;
4) Solve the factorized game graph via backward induction.

We will see in Sec. V that this procedure can deliver enormous advantages, for example, in the autonomous driving domain.

We are left with assessing when and how the intersection of resources among players is a good factorization heuristic.

Assumption 1. *The following implications must hold for resources to serve as a factorization tool. Let* $i, j \in \mathcal{A}$ *with respective state* x_i, x_j

- *Empty intersection of reachable resources implies strategic independence between* i, j*:*

$$\mathsf{futureRes}(\mathbf{G}(x_i)) \cap \mathsf{futureRes}(\mathbf{G}(x_j)) = \emptyset \Rightarrow i \perp\!\!\!\perp j.$$

- *Empty intersection of reachable resources under optimal solutions implies independence at equilibrium between* i, j*.*

$$\mathsf{futureRes}^{\gamma_i^*}(\mathbf{G}(x_i)) \cap \mathsf{futureRes}^{\gamma_j^*}(\mathbf{G}(x_j)) = \emptyset$$

$$\forall \gamma_j^*, \gamma_j^* \in \textsc{Sol}(\mathbf{G}(x_i)) \times \textsc{Sol}(\mathbf{G}(x_j)) \Rightarrow i \perp\!\!\!\perp^* j.$$

At the practical level, Assumption 1 requires that the state-to-resources projection (Res) captures all the interaction costs between the agents. For instance, a proximity cost of an agent needs to be captured by its occupancy map (e.g., by inflating its state occupancy).

B. Factorization algorithms over future resources

We introduce two factorization algorithms that leverage the reachable resources computed solving the individual games.

FACT1 factorizes building the strategic influence graph for each new candidate state. Consider the expansion of a given game node. For each action profile we find a new

[3]Here we see a parallel with pathfinding algorithms.

candidate state ($\hat{x}$) of the players. For each pair of players we know that they could be strategically dependent if the reachable future resources intersect (line 3-5 Alg. 2). Subsets of players that are independent of each others will eventually have no connection. Thus, the connected components of $\mathcal{I}$ represent the factorization of the new state. The new state gets factorized in as many new game nodes as the number of connected components. This process is repeated while building the game graph, for each game node to be expanded, for each feasible joint action profile.

FACT2 is a refinement of FACT1 based on the observation that we can try to factorize single players based on their *optimal* future resources. The ones associated to the their solution strategy for the alone game. This is often a much smaller set than all the reachable ones. For partitions with more than one player we still have to use the *reachable* future resources. These provide an upper bound to the resources required by the solution of that subgame. Recall indeed, that with the proposed approach, when we build the game graph we know only the solutions (hence the required resources) of the individual (sub)games. The algorithmic summary of FACT2 is provided in Alg. 3.

Algorithm 2 FACT1 algorithm

Input $\mathsf{futureRes}(\mathbf{G}(\hat{x}_i)) \quad \forall i \in \mathcal{A}$

1: **function** FACT1($\hat{x}$)
2: $\mathcal{I} = \langle \mathcal{A}^{\hat{x}}, \{\} \rangle$ ▷ The influence graph
3: **for** $(i,j) \in$ COMBINATIONS($\mathcal{A}^{\hat{x}}, 2$) **do**
4: **if** $\mathsf{futureRes}(\mathbf{G}(\hat{x}_i)) \cap \mathsf{futureRes}(\mathbf{G}(\hat{x}_j))$ **then**
5: $\mathcal{I}$.add_edge((i,j))
6: **return** CONNECTED_COMPONENTS($\mathcal{I}$)

Algorithm 3 FACT2 algorithm

Input $\mathsf{futureRes}(\mathbf{G}(\hat{x}_i))$, $\mathsf{futureRes}^{\gamma_i^*}(\mathbf{G}(\hat{x}_i)) \quad \forall i \in \mathcal{A}$

1: **function** FACT2($\hat{x}$)
2: $\mathcal{I} = \langle \mathcal{A}^{\hat{x}}, \{\} \rangle$ ▷ The influence graph
3: partitions $\leftarrow \{\}$ ▷ Current agents' partition
4: **while** partitions != CONNECTED_COMP($\mathcal{I}$) **do**
5: partitions = CONNECTED_COMP($\mathcal{I}$)
6: **for** $\{i\} \in$ partitions **do**
7: **if** len($\{i\}$) == 1 **then**
8: $\text{futureRes}_{\{i\}} \leftarrow \mathsf{futureRes}^{\gamma_i^*}(\mathbf{G}(\hat{x}_i))$
9: **else**
10: $\text{futureRes}_{\{i\}} \leftarrow \cup_{i \in \{i\}} \mathsf{futureRes}(\mathbf{G}(\hat{x}_i)$
11: **for** $(\{i\},\{j\}) \in$ COMBINATIONS(partitions,2) **do**
12: **if** $\text{futureRes}_{\{i\}} \cap \text{futureRes}_{\{j\}}$ **then**
13: $\mathcal{I}$.add_edges($(\{i\},\{j\})$)
14: **return** partitions

V. EXPERIMENTAL RESULTS AND DISCUSSION

We exemplify factorization capabilities in driving scenarios. The players are AVs solving the motion planning problem finding an equilibrium policy for the game. The proposed setup is integrated on reproducible, publicly available CommonRoad scenarios [28].

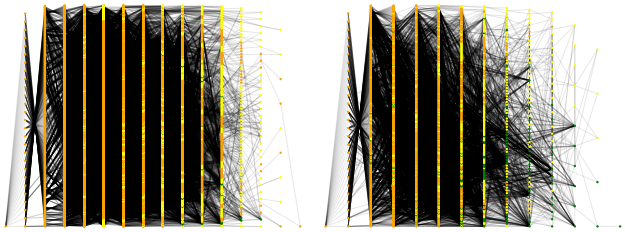

Fig. 3. The game graph resulting from the 3-player case (played from left to right). Nodes represent game nodes, edges the possible transitions. On the left, NAIVEDG has 18607 nodes. On the right FACT2 with 3593 nodes. One can appreciate graphically how the graph on the right results with many fewer nodes, fewer edges, and more yellow/green nodes. The color encodes the number of players in the game node: orange 3 players, yellow 2, green 1. Note that some yellow/green nodes are present also on the left due to early termination of the other players.

The state of each player is described by its progress along its reference lane, its current speed, and the waiting time (how long it has been waiting in that position). The vehicle's action space corresponds to a discrete set of feasible accelerations. Each player has the objective of minimizing the time to destination while not colliding and minimizing the safety distance violation from other players. We express these objectives as a lexicographic preference over the outcomes. We ensure to write the players' preference satisfying the communal properties of Urban Driving Games (UDGs) [29], thus making the game potential. This allows us to search for equilibria in pure strategies, with also the guarantee that the global optimum minimizes the social cost.

A. Case study

We consider a multi-lane intersection (USA_Lanker from [28]) with up to 5 vehicles as the testbed for factorization. Each vehicle has a starting position and a goal progress. These are visible in the first subfigure of Fig. 4 along with the resources associated to the initial state. In the same setup we introduce additional players one at a time. Hence, as summarized in Table I, we end up with 4 scenarios where, respectively, only the first 2 players are present, then only the first 3, and so on.

Each of the scenarios is solved for the same identical initial conditions by three solvers. 1) NAIVEDG solves the dynamic game without factorization, builds the game tree directly on the state space of all the players and solves it via backward induction; 2) FACT1 solves first the single DPs of the players collecting the resources, then builds the factorized game graph using Alg. 2, finally solves it via backward induction; 3) FACT2 works as FACT1 but factorizes according to Alg. 3. A graphical illustration of the resulting game graph is shown in Fig. 3 for the 3-player case.

B. Implementation details

Cost functions: The stage cost of each player is a lexicographic cost minimizing: 1) collision energy, 2) safety distance violation, 3) time. Note that not all monoids on such a cost structure would preserve the Bellmann optimality

TABLE I

SUMMARY OF SOLVE TIMES AND GAME NODES NUMBERS.

Players	2		3		4		5	
	t [s]	# Nodes	t [s]	# Nodes	t [s]	# Nodes	t [s]	# Nodes
NAIVEDG	0.7	729	26.7	18607	1568	465567	OOM[a]	OOM[a]
FACT1	0.85 (+21%)	439 (-40%)	11.5 (-57%)	4386 (-76%)	63.5 (-96%)	12474 (-97%)	989	68586
FACT2	0.82 (+17%)	434 (-40%)	8.83 (-67%)	3593 (-81%)	54.2 (-97%)	10041 (-98%)	591 (-40%)[b]	33685 (-51%)[b]

[a] Out of memory.
[b] Relative gain with respect to FACT1.

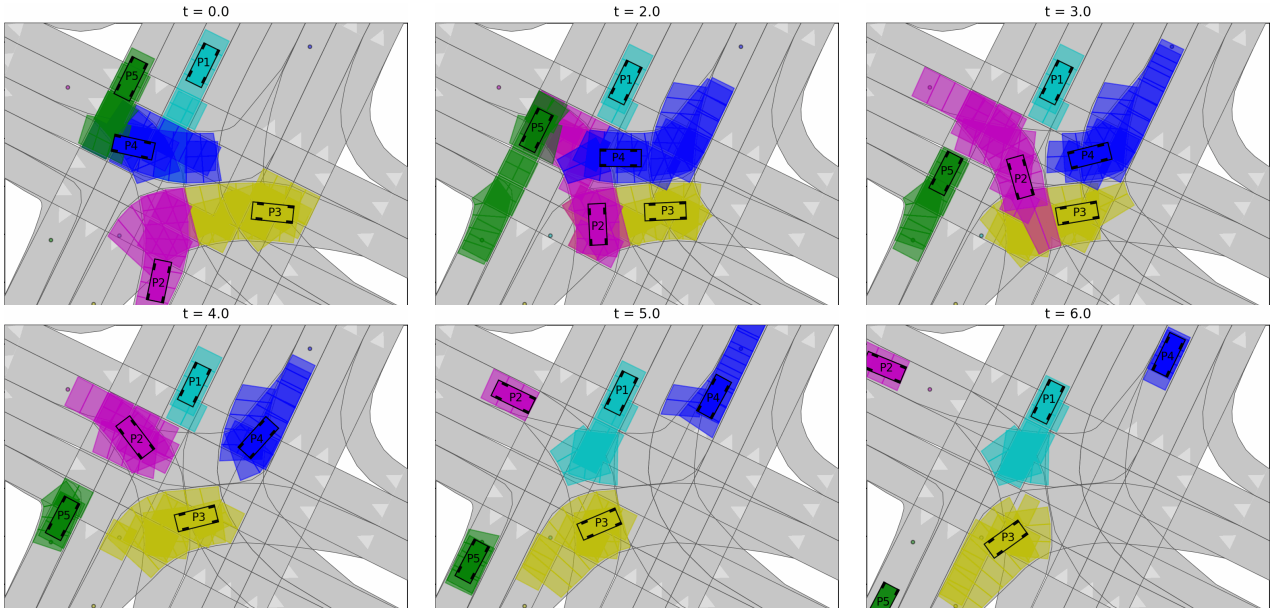

Fig. 4. A visualisation of the 5-player case with its equilibrium solution. Each player is visualized with the resources associated to its current state. One can observe that they span asymmetrically more to the front of the vehicles. This occurs because they need to "upper bound" the stage costs such that an empty intersection of resources implies no joint stage costs between the players when transitioning to the next state. As a result, a vehicle travelling at higher speed requires more resources. Finally, recall that for factorization the independence check involves not only the visualized resources but also all the ones associated to the future reachable states.

principle. For example, propagating only the maximum violation of safety distance. We propagate the stage costs with a cumulative sum which preserves the optimality principle and allows us to solve the game graph in a single pass via backward induction.

Discretization: The action set of the vehicles is discretized into 3 possible acceleration values ($\{-1, 0, +1\}$). We remove negative accelerations if they cause negative velocities. The position of the vehicles is quantized to the three most significant digits of progress (a progress of 15.43 and 15.46 are the same state). The stage discretization is a parameter set to 2 [s] for the reported results.

Terminal condition: Currently for a player the game terminates only if it reaches the goal progress or it collides and no another active player is in its game node. To avoid players waiting at a certain state forever we set a maximum waiting time (1 stage), after that, the only action available to the vehicle are positive accelerations. This parameter has been set arbitrarily to have the game terminating within reasonable depth of the tree in any case. We imagine that most of these precautions can be avoided setting a maximum lookahead in number of stages.

Resources: The set of available resources is obtained by subdividing the lanelets of the road network in polygons with a length of 1.5 m for the reported results. A more granular subdivision would allow a less conservative factorization.

Solutions: We solve the game graphs exhaustively finding closed loop equilibria for all the game nodes. An exception is made for FACT2, whose factorization depends on the solutions of the individual DPs. Thus, not considering the possibility for an "alone" player to deviate from the optimal strategy. For now, we delegate the study of the specific factorization algorithms tailored to specific solution concepts to future work.

C. Results and Discussion

We report in Table I the summary results for the presented study cases. We compare FACT1 and FACT2 with a baseline that does not use any factorization (NAIVEDG). The computation times include creation of all the game graphs, resource computation (when needed) and solutions. Similarly, for the factorization algorithms, the number of nodes includes also the ones generated in individual games. While the values are still in the same order of magnitude for the 2-player case, already by just adding an additional player this is not the

case. For the 4-player case, NAIVEDG generates almost half a billion nodes versus the factorized counterparts that go slightly above $10k$.

Comparing FACT1 and FACT2, we observe that, for the cases at hand, it is worth the extra effort in factorization checks. In particular, FACT2 is able to factorize earlier single players into "alone" games. This is particular evident for the 5-player case. Indeed P5 (in green in Fig. 4), has a starting condition that is prone to early factorization in the game graph. In particular when considering its future *optimal* resources, it quickly becomes independent of the others. This does not happen so "early" with FACT1 that considers its reachable resources, hence the possibility of slowing down and still interfering with P2 (in magenta in Fig. 4).

Even though the relative gains and the number of nodes are what matters the most in comparing the algorithms, we believe that also the absolute numbers in solving time can be significantly improved. The current implementation uses Python as programming language, and includes realistic cost structures that could be slimmed down if needed (e.g., by not computing collision energy of collisions). We do believe that an implementation in strongly typed languages with minimal cost structures could also reduce the absolute solving time of $2-3$ orders of magnitudes. Yet, as the number of players grows, it would show asymptotically the same relative gains.

VI. CONCLUSION AND OUTLOOK

We introduced *factorization* of a dynamic game as a transformation of the original game that is strategically equivalent but much simpler to solve. We presented two algorithms for factorization that are applicable to dynamic games in the robotic context. We further imagine that many other exists also in different application domains. Indeed, we see a parallel with heuristics of path finding algorithms. Some algorithms for factorization can be more suited to certain types of games than others. In this sense, we conceive to evaluate a factorization algorithm as a trade off between computation overhead and degree of the factorization. Finally, we envision the possibility of including asymmetric dependence of players. In the presented domain of autonomous vehicles this could, for example, depend on traffic rules (e.g., right of way).

REFERENCES

[1] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2 2022.

[2] D. Koller and B. Milch, "Multi-agent influence diagrams for representing and solving games," *Games and Economic Behavior*, vol. 45, no. 1, pp. 181–221, 10 2003.

[3] M. R. Levy and J. Tasoff, "Exponential-growth bias and overconfidence," *Journal of Economic Psychology*, vol. 58, pp. 1–14, 2017.

[4] S. Le Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: a fast augmented Lagrangian solver for constrained dynamic games," *Autonomous Robots*, vol. 46, no. 1, pp. 201–215, 1 2022.

[5] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots*, vol. 42, no. 7, 10 2018.

[6] A. Dreves and M. Gerdts, "A generalized Nash equilibrium approach for optimal control problems of autonomous cars," *Optimal Control Applications and Methods*, vol. 39, no. 1, pp. 326–342, 1 2018.

[7] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical Game-Theoretic Planning for Autonomous Vehicles," *2019 International Conference on Robotics and Automation (ICRA)*, vol. 2019-May, pp. 9590–9596, 10 2018.

[8] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, S. L. Herbert, S. Wang, A. D. Dragan, and C. J. Tomlin, "Confidence-aware motion prediction for real-time collision avoidance," *International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 250–265, 2020.

[9] R. Tian, N. Li, I. Kolmanovsky, Y. Yildiz, and A. R. Girard, "Game-Theoretic Modeling of Traffic in Unsignalized Intersection Network for Autonomous Vehicle Control Verification and Validation," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2020.

[10] G. B. Davis, M. Benisch, K. M. Carley, and N. M. Sadeh, "Factoring games to isolate strategic interactions," *Proceedings of the International Conference on Autonomous Agents*, no. January, pp. 443–449, 2007.

[11] A. Zanardi, G. Zardini, S. Srinivasan, S. Bolognani, A. Censi, F. Dorfler, and E. Frazzoli, "Posetal Games: Efficiency, Existence, and Refinement of Equilibria in Games With Prioritized Metrics," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1292–1299, 4 2022.

[12] M. Kearns, "Graphical games," *Algorithmic Game Theory*, pp. 159–180, 2007.

[13] A. X. Jiang, K. Leyton-Brown, and N. A. Bhat, "Action-Graph Games," *Games and Economic Behavior*, vol. 71, no. 1, pp. 141–173, 2011.

[14] L. Zhou, K. Lü, and W. Liu, "Game theory-based influence diagrams," *Expert Systems*, vol. 30, no. 4, pp. 341–351, 2013.

[15] R. D. Shachter, "Decisions and dependence in influence diagrams," *Journal of Machine Learning Research*, vol. 52, no. 2016, pp. 462–471, 2016.

[16] D. Geiger, T. Verma, and J. Pearl, "Identifying independence in bayesian networks," pp. 507–534, 1990.

[17] R. D. Shachter, "Bayes-Ball: The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams)," *CoRR*, pp. 480–487, 2013.

[18] J. Q. Smith and P. E. Anderson, "Conditional independence and chain event graphs," *Artificial Intelligence*, vol. 172, no. 1, pp. 42–68, 2008.

[19] P. A. Thwaites and J. Q. Smith, "A graphical method for simplifying Bayesian games," *Reliability Engineering and System Safety*, vol. 179, pp. 3–11, 2018.

[20] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2017.

[21] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory, 2nd Edition*, SIAM, Ed. Society for Industrial and Applied Mathematics, 1 1998, vol. 85, no. 3.

[22] A. Haurie, J. B. Krawczyk, and G. Zaccour, *Games and Dynamic Games*, ser. World Scientific-Now Publishers Series in Business. World Scientific, 3 2012, vol. 1.

[23] K. Slutsky, D. Yershov, T. Wongpiromsarn, and E. Frazzoli, "Hierarchical Multiobjective Shortest Path Problems," in *Algorithmic Foundations of Robotics XIV*, S. M. LaValle, M. Lin, T. Ojala, D. Shell, and J. Yu, Eds. Cham: Springer International Publishing, 2021, pp. 261–276.

[24] J. P. Hespanha, *Noncooperative game theory : an introduction for engineers and computer scientists*. Princeton University Press, 2017.

[25] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," *arXiv:1911.10635 [cs.LG]*, pp. 1–72, 2019.

[26] J.-S. Pang and G. Scutari, "Nonconvex Games with Side Constraints," *SIAM Journal on Optimization*, vol. 21, no. 4, pp. 1491–1522, 10 2011.

[27] M. Althoff, G. Frehse, and A. Girard, "Set Propagation Techniques for Reachability Analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 369–395, 5 2021.

[28] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 719–726, 2017.

[29] A. Zanardi, E. Mion, M. Bruschetta, S. Bolognani, A. Censi, and E. Frazzoli, "Urban Driving Games With Lexicographic Preferences and Socially Efficient Nash Equilibria," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4978–4985, 7 2021.