An algorithm for multi-robot planning: SGInfiniteVI

Mohamed Amine Hamila, Emmanuelle Grislin-Le Strugeon, René Mandiau, Abdel-Illah Mouaddib

An algorithm for multi-robot planning: SGInfiniteVI

Mohamed Amine Hamila[1,2,3], Emmanuelle Grislin-le Strugeon[1,2,3], Rene Mandiau[1,2,3], Abdel-Illah Mouaddib[4]
[1] *Univ Lille Nord de France, F-59000 Lille, France*
[2] *UVHC, LAMIH, F-59313 Valenciennes, France*
[3] *CNRS, UMR FRE 3304, F-59313 Valenciennes, France*
[4] *GREYC, Universite de Caen Basse-Normandie, France*
{*mohamed.hamila,emmanuelle.grislin,rene.mandiau*}*@univ-valenciennes.fr*
mouaddib@info.unicaen.fr

***Abstract*—In this paper we introduce an efficient path planning algorithm called SGInfiniteVI so the resulting path solution facilitates for robots the accomplishment of their tasks. The algorithm is based on the dynamic programming and deals with the problem of distributed equilibrium computation and selection in general-sum stochastic games with infinite horizon. Our approach proposes three Nash selection functions (in case of multiplicity of Nash equilibria) and obtains its benefits by pointing out experimentally which function is the most suitable to favor better interaction between robots. Experiments were conducted on the classical box-pushing problem, with several mobile robots moving objects in dynamic environments.**

***Keywords*-Stochastic Games; Multi-agent Planning; Markov processes; Game theory;**

I. Introduction

To design and deploy a multi-robot system constitutes an important research challenge. Indeed, mobile robots have become autonomous enough, so that we have to think of their organizations. We are considering in this study how robots can render service by moving by themselves. Thus, multiple practical applications can be considered, such as the exploration, search and rescue, people rescue, etc. In this kind of systems, the mutual interactions between individual robots sharing the same workspace are complex in general cases and one of the most important issues is how to coordinate the behavior of each robot so that the overall performance can be optimized. In order to facilitate tasks achievement, the common approach consists of time and continuous-space discretization into a finite collection of cells. The problem is then reduced to decide which cell to visit at each time interval. This looks like the class of multi-agent sequential decision problems under uncertainty [1], where it is necessary at each step to decide what joint action to choose.

The framework of "Stochastic Games" seems to be a convenient tool for modeling and solving multi-robot interaction problems. It provides a theoretical basis for many recent works [2], [3], [4], [5], dealing with planning and learning in multi-agent sequential decision making. Stochastic games can be considered as an extension of Markov Decision Process (MDP) [6], [7] and game theory [8], [9]. It is possible to have multiple agents (that may be non-cooperative), whose actions directly impact the resulting rewards and the transition to the next state.

In this paper we focus on planning approaches for solving stochastic games. There are few studies that address this problem [4], [10]. The most recent work, by Kearns et al. [4], proposes the FINITEVI algorithm. However, this algorithm requires a high level of centralization, and because of multiplicity of Nash [11] equilibria its direct application remains an open problem. Thus the main contribution of our work is actually twofold:

- Firstly, we propose a novel algorithm for planning in general-sum stochastic games with an infinite horizon, called SGInfiniteVI[1], taking into account the necessity of decentralization and offering different approaches for the selection of Nash equilibria. To prove in practical its efficiency, we tested SGInfiniteVI algorithm on a box-pushing problem. We show empirically that under certain conditions robots can often select the same Nash equilibrium and therefore choose non-conflicting actions.
- Secondly, we focus on the robustness of the Nash equilibrium against the non-cooperation of one of the robots. In this case, several scenarios can be investigated, as the failure of a robot or its substitution by a human. The result is promising and opens up interesting prospects as the construction of joint-decisions between robot and human [12], [13].

The paper is organized into four sections. Section 2 recalls the definitions and notations of stochastic games and their solving methods. Section 3 describes our SGInfiniteVI algorithm and shows how to apply the model of stochastic games to a grid-world game. Some preliminary results are presented and discussed in section 4 and finally we conclude in section 5.

II. Stochastic games

In this section we present an introduction to stochastic games, and then two crucial aspects of the model, namely

[1]SGInfiniteVI is short for "Stochastic Games Infinite Value Iteration".

the Nash equilibrium and the resolution methods.

A. Definitions and concepts

Stochastic Games (SG) [14] are defined by the tuple:

$$< Ag, \{A_i : i = 1 \ldots |Ag|\}, \{R_i : i = 1 \ldots |Ag|\}, S, T >$$

- Ag: finite set of agents.
- A_i: finite set of actions of agent i ($i \in Ag$). A pure strategy corresponds to an action chosen in a deterministic way.
- R_i: reward function of the player i, $R_i(a) \rightarrow \mathbb{R}$, where $a =< a_1, a_2, \ldots, a_{|Ag|} >$ $(a \in A)$.
- S: finite set of states of the game.
- T: transition function, $T : S \times A \times S \rightarrow [0, 1]$, indicating the probability of moving from a state $s \in S$ to a state $s' \in S$ by running the joint action a.

The stochastic games extend MDPs [6] to several agents, selecting simultaneously different actions. Henceforth the next state and the rewards depend on actions of all agents. Each agent has a separate reward function and the main objective is to select actions that maximize its future gains.

These games can also be seen as an extension of game theory but with multiple states. In game theory, the set of all possible gains that can be obtained by two agents is generally represented by a two-dimensional matrix. The choices jointly made determine the gains of each of the agents according to the played matrix.

The particularity of stochastic games is that each state s can be considered as a matrix game $M(s)$. For example, suppose that both players start the game in state s, then they play the matrix $M(s)$. Immediately after, players receive a payment. The game then goes to $M(s')$ with probability $P(s'|s, a)$ depending on the joint-action a played at step s and so on.

B. Nash equilibrium in stochastic games

As in a Markov Decision Process, each agent computes its own policy that maximizes performance criterion. This criterion is usually the Expected discounted sum of its future rewards: $E\ [\ \sum_{t=0}^{\infty} \gamma^t R^t\ ]$.

However, general-sum stochastic games have reward functions which can be different for every agent. In certain cases, it may be difficult to find policies that maximize the performance criteria for all agents. That is why in general-sum stochastic games, equilibrium is always looked for. This equilibrium is a situation in which no agent can improve its performance criteria if it is the only one to change policy: we find here the definition of Nash equilibrium.

Definition 1 *A Nash Equilibrium is a set of strategies (actions) a^* such that:*

$$R_i(a_i^*, a_{-i}^*) \ \geq \ R_i(a_i, a_{-i}^*)\ \forall i \in Ag,\ \forall a_i \in A_i \tag{1}$$

The concept of Nash equilibrium can also be written using the concept of Best Response:

Definition 2 *Given the other players' actions a_{-i}, the Best Response (BR) of the player i is:*

$$BR_i : a_{-i} \rightarrow argmax_{a_i \in A_i}\ R_i(a_i, a_{-i}) \tag{2}$$

thus, we say that a^* is a Nash equilibrium if:

$$a_i^* \ \in \ BR_i(a_{-i}^*)\ \ \forall i \tag{3}$$

Definition 3 *The set of all equilibria, one per state, constitutes a "joint-policy":*

$\Pi : S \rightarrow A,\ A\ \in \{A_1 \times A_2 \times ... \times A_{|Ag|}\}$

This policy can also be considered as a vector $\Pi = (\pi_1, \pi_2, ..., \pi_{|Ag|})$ of an individual agent policies.

The Nash equilibrium is often used in general-sum stochastic games as the equilibrium of better mutual answer (while ensuring a better response for their individual goals). Thus, when all other agents follow the joint-policy Π, the performance criteria for the agent i will be: $E\ [\ \sum_{t=0}^{\infty} \gamma^t R_i^t(\Pi)\ ]$.

However, the Nash equilibrium is not always unique and the fact to have (for the same game) multiple equilibria with different values generally poses coordination problems. Indeed, in this case if agents choose to play different equilibria, the joint-action performed may not be an equilibrium. The next section will show how planning techniques will address these problems.

C. Solving stochastic games

In this section we will discuss some solutions for stochastic games. The main works concerning the planning in stochastic games are the ones of Shapley (1953) and Kearns et al. (2000). The first one presents an algorithm finding a Nash equilibrium by planning in zero-sum[2] stochastic games. The second bases itself on the first one and widens it to general-sum stochastic games but with finite horizon.

1) Algorithm of Kearns et al: our work was inspired by Kearns et al. [4] dealing with planning in stochastic games with finite horizon and two agents. We are particularly interested by their algorithm called FiniteVI (algorithm 1) which finds a centralized Nash equilibrium of a stochastic game with a horizon T. The algorithm is based on the principle of value iteration, computes a value function and converges by successive iterations to the optimal solution with the following Bellman equation [15]:

$$V(s) = R(s) + max_a \sum_{s' \in S} T(s, a, s')\ V(s')$$

The algorithm loops over the horizon T and at each step t computes for every state s of the stochastic game the correspondent matrix (line 4 of Algorithm 1), then selects a unique Nash equilibrium (line 6 Algorithm 1) using the function f, the value of the equilibrium being given by V_k^f. The arbitrary function f[3], introduced by Kearns et al. in their

[2]zero-sum: describes a game in which the gains of one are the losses of the other.

[3]According to Kearns et al., recall that f is an arbitrary Nash Selection Function, f_k extracts the strategy selected by the player k.

article, chooses a Nash equilibrium and provides for each agent the correspondent policy to play through the functions f_1 and f_2. The authors have shown that the complexity of their algorithm is linear on the horizon time but quadratic in terms of state space size.

Algorithm 1 Algorithm FiniteVI

Require: A general-sum stochastic game SG

1: **for all** t=1...T **do** {time steps}
2: **for all** $s \in S, k \in 1, 2$ **do** {state s and player k}
3: **for all** $a \in A$ **do** {joint-action a}
4: $M_k(s,t) = R_k(s,a) + \sum_{s' \in S} T(s,a,s')\, V_k^f(M_1(s',t-1),\ M_2(s',t-1))$
5: **end for**
6: $\pi_k(s,t) = f_k(M_1(s,t), M_2(s,t))$
7: **end for**
8: **end for**
9: **return** The policy pair (π_1, π_2)

2) Discussion: we observe that the FiniteVI algorithm requires a high level of centralization: the presence of the function f during the planning process to select the same equilibrium for all agents. Moreover, this function f was not clearly defined and may be considered as an "oracle". We think that an implementation of the above algorithm is an open problem.

Our work is in the spirit of this approach and presents a decentralized planning algorithm for general-sum stochastic games with infinite horizon. It aims to:

- Preserve the advantage of the planning which allows us to find a better solution more quickly than the algorithms of learning as far as we have the model of the world.
- Get rid of the centralization of the algorithm proposed by Kearns et al.
- Encode the Nash selection function f.
- Seek and find the best Nash selection function in case of multiplicity that will minimize conflicts and livelocks during the execution phase (simulations).
- Compute a stationary policy (infinite horizon): we adopt an empirical approach aimed to adapt the algorithm presented previously to the case of an infinite horizon.

III. A NEW ALGORITHM AND APPLICATION ON A GRID-WORLD GAME

In this section, we present our algorithm SGInfiniteVI, and then we will apply the model of S.G. to a scenario of interaction between robots.

A. Proposal of an algorithm

As mentioned above, the main objective of this work is to avoid the centralization implied by the selection function f. An intuitive idea consists in distributing the computation of equilibria. This supposes that each agent will enjoy of its own Nash selection function but also equilibria calculated at each step may differ from one agent to another. Since the problem arises only in cases where several Nash equilibria coexist for the same game, we will try to find in such circumstances the best method that will lead all agents to choose the same equilibrium.

SGInfiniteVI (algorithm 2) takes as parameters: a stochastic game SG, γ (discount factor), ϵ (stopping condition) and returns policy actions for the player.

1) Differences with FiniteVI: We can summarize our contribution with regard to the previous algorithm in three important points:

- Decentralization and Nash selection function: the best response functions can be used to determine the function f. This suggests a procedure for finding a Nash equilibrium:
  1) To find the best response function of each player (equation 2).
  2) To identify a set of actions for which both players play their best response (equation 3).

  This method guarantees to identify all Nash equilibria which can exist in the game. Indeed in case of multiplicity of Nash equilibrium, several variants of implementation of the function f are available. It is possible to choose between multiple equilibria, for example: to maximize the global/individual payoff or to choose a non-Pareto-dominated Nash[4]. It will be possible to show the variant which will (at best) avoid problems of coordination between agents during simulations. Let us note also that each agent will perform the same instantiation of the algorithm but this does not mean that the equilibria calculated by each are the same for the others.
- Number of agents > 2: to increase the number of agents, we must now build a multidimensional array instead of a simple two-dimensional matrix. It is an $|Ag|$-dimensional hypercube, where each dimension represents the decisions for the involved player.
- Infinite horizon: to adapt the algorithm to the case of infinite horizon and with reference to the classical dynamic programming algorithms, it is necessary to add the discount factor γ which incorporated into the matrices computed at each step (line 5 of algorithm 2) and the stopping condition ϵ (line 9 of algorithm 2).

2) SGInfiniteVI Complexity: For searching equilibria, our algorithm is dealing only with a pure Nash equilibrium. It has been proved that even with very restrictive conditions on the players strategies, to determine if a game has a "pure

[4]non-Pareto-dominated Nash: it is a Nash equilibrium non-Pareto-dominated by another Nash equilibrium. Also, the game is in a Pareto optimal state if no player can increase its profit without damaging the gain of at least one other.

Algorithm 2 Algorithm SGInfiniteVI

Require: A general-sum stochastic game SG, $\gamma \in [0,1]$, $\epsilon \geq 0$

$t \leftarrow 0$
repeat
  for all $s \in S, k \in 1 \ldots |Ag|$ **do**
    for all $a \in A$ **do**
      $$M_k(s,t) = R_k(s,a) + \gamma \sum_{s' \in S} T(s,a,s')\ V_k^f(M_1(s',t-1),\ldots,\ M_{|Ag|}(s',t-1))$$
    end for
    $\pi_k(s,t) = f_k(M_1(s,t), \cdots,\ M_{|Ag|}(s,t))$
  end for
until $max_{s \in S} |V_k^f(s,t) - V_k^f(s,t-1)| < \epsilon$
return The policy π

Nash equilibrium" is NP-complete [16], [17]. Therefore, the running time of SGInfiniteVI algorithm is not polynomial in the size of the game matrix (due to the fact that the function f used to compute Nash equilibrium is itself non-polynomial). Moreover, the sizes of state space and joint action are exponential in the number of agents. Overall, we believe that the running time of our algorithm is exponential.

As for the space complexity, it mainly depends on the payoff matrix that will be generated every time that the corresponding state is valued. The matrix size is static: $|A_1| \times |A_2| \times ... \times |A_{|Ag|}| = |A|$.

Each entry in the matrix is composed of $|Ag|$ values, making a total of $|Ag| \times |A|$ values. Every agent must maintain a backup matrix at each state s, including its own payoff but also the payoffs of the other agents. Therefore the number of matrices stored during algorithm iterations is $|S|$ and the total number of required values is: $|Ag| \times |S| \times |A|$. SGInfiniteVI is linear in the number of states but exponential in the number of agents.

Fortunately, when the algorithm converges, there is no longer need to keep matrices in memory and we get back only a policy (state-action pairs).

B. A grid-world game

The example that we choose is similar to the examples from literature, as the "Two-Player Grid Soccer" of Littman [18] or the "Two-Player Coordination Problem" of Hu and Wellman [19], but a little more difficult. It is the problem of multi-robot box-pushing (see figure 1), the game includes: robots, objects and a container box. For example, the authors of [19] study Q-learning as a way to solve this application; but, as in their analysis, our works consider the following hypothesis: "one game has deterministic moves, and the other has probabilistic transitions" [19].

1) Description: In this example, the objective of the robots is to put all the objects in the box with a minimum number of steps. Robots have no preference on the objects to be carried, do not communicate, do not share any initial plan and when they must make decisions they must avoid any conflict (to take up the same place, to carry the same object). Despite the simplicity of the problem, grid-world games possess all the key elements of dynamic games: location- or state-specific actions, state-transitions, and immediate and long-term rewards.

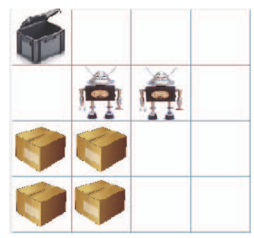


Figure 1. An example of scenario: two robots, four objects and a container box.

2) Modeling as a stochastic game: The game presented above can be easily modeled through a stochastic game. Let us recall that the model of S.G. is defined by the tuple:

$$< Ag, \{A_i : i = 1 \ldots |Ag|\}, \{R_i : i = 1 \ldots |Ag|\}, S, T >$$

- Ag: two or three robots (we chose to limit the number of robots to three, mainly for problems of computation time).
- A_i: each robot has a set of five actions: {(U)p, (D)own, (L)eft, (R)ight, (N)othing}.
- R_i: the reward function is defined in the same way for all robots, with the following arbitrary values:

$$\begin{cases} +100 & \text{when all objects are transported into the container box.} \\ -200 & \text{if the robot occupy the same position of another one.} \\ -200 & \text{if the robot carries an object and try to carry another.} \end{cases}$$

- S: this is the set of states consisting of:
  - Geographic positions of robots/objects in the grid.
  - Internal state of a robot (carrying or not an object).
  - Internal state of an object (carried or not).
- T: this function represents the response of the environment to the executed actions. It will be defined according to these cases:
  - A robot cannot carry two objects at a time.
  - If a robot moves in a cell containing an object then it must carry this object.
  - If an action brings an object into the container box, then this object is removed from the grid and the robot (which transported it) is marked as empty.
  - Robots move on a toroidal grid.

The following section intends to validate our algorithm on the modeled game.

IV. Validation

In this section, we present the results obtained in applying our SGInfiniteVI algorithm. Firstly we will start by showing

the mode of operation of the algorithm (to ease in its centralized version) according to the game previously presented. Then we measure the computation time and memory space required by the algorithm and we show that the threshold of one million states is reasonably calculable. Finally we will see through its decentralized version the best Nash selection function that will facilitate coordination and minimize conflicts situations between agents. The simulator has been implemented in the JAVA language and the experiments were done on a machine quad core 2.8 GHZ and 4 GB of memory.

A. Observation on the convergence

Along the experiments we observed that the algorithm has always converged consistently and quickly, despite the fact that it is not guaranteed to do so. Recently, Zinkevich et al. [20] showed that the Value-Iteration algorithm may not converge in certain case of general-sum stochastic games with infinite horizon, but converges always to cyclic equilibrium policies. Indeed, the Nash equilibrium operator is usually not a contraction operator. However, Hu and Wellman [19] have shown that it is a non-expansion operator and that under restrictive conditions convergence is guaranteed. Alongside their work, we believe that the convergence of such algorithms depends strongly on the structure of matrix games evaluated during planning and therefore it is not fair to generalize convergence to the full range of stochastic games.

B. Evaluation on computation time and memory space

Despite the simplicity of the example we can note that the size of the state space is not negligible. The complexity is: $(2\ n^2)^{|Ag|}\ *\ 2^O$ where $|Ag|$ represents the number of agents, O is the number of objects and n^2 is the grid size. For example, for a grid size of 8 (64 cells), 3 agents and 3 objects the total number of states is $16,777,216$;

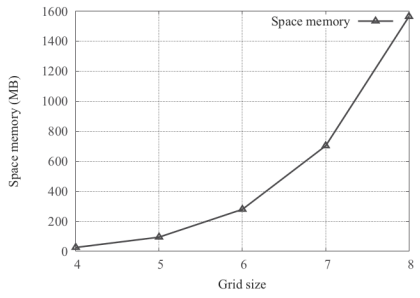


Figure 2. Memory space required to compute a policy with three agents, three objects and different grid sizes (n $\times$ n).

We noted that the construction of matrices and the search for equilibria increase considerably the size of needed memory (figure 2) and the computation time (figure 3). It seems that even with a large state space, memory requirements remain reasonable and only the computation time policy

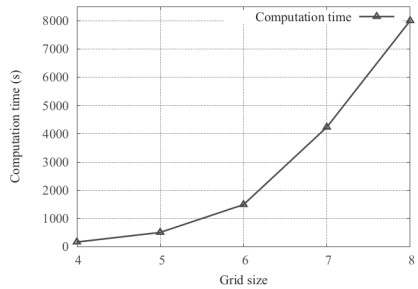


Figure 3. Computation time required to compute a policy with three agents, three objects and different grid sizes (n $\times$ n).

seems a priori problematic. We have not considered this problem in our work; but to decrease the CPU time we know that exist heuristics to reduce the size of matrices and to improve the method of searching equilibria [21], [22].

C. Principle

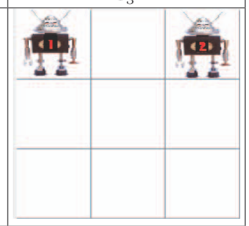
SGInfiniteVI algorithm provides a policy for each agent. More precisely, the agent plans actions based on its own payoff matrix. But, due to the lack of space, we present only an illustration of the centralized version of SGInfiniteVI algorithm (with a single payoff matrix for all agents).

The following example (table I) shows how two agents manage together to coordinate themselves, to avoid collisions and to achieve their basic objective. The game starts from an initial state chosen randomly. After observing the current state, agents choose their actions simultaneously and thus they discover the new state. The game stops when all objects are placed in the container box. Each state of the world is represented with a "screen-shot", a payoff matrix[5] and the future Nash-actions to do. Payoffs representing the selected Nash equilibrium are marked in a gray cell. The steps are:

- Step 1: *robot 2* goes to the right (to take an object) and *robot 1* moves to approach the remaining object. The equilibrium chosen in this step is important because it prevents the competition on the objects to be transported and defines the mission of each robot.
- Step 2: this equilibrium confirms the rationality of robots. Indeed, if *robot 2* tries to maximize its gains then it will never go to a square where there is an object. Therefore, *robot 1* can carry safely the remaining object with the certainty that the other robot cannot take it.
- Step 3: *robot 2* yields the way to allow *robot 1* to put the object that it was carried.

Note that this centralized approach guarantees that there is no conflict between robots but it requires a high degree of centralization. The next section will attempt to present the main contribution of this work namely the decentralized version of SGInfiniteVI algorithm.

[5]Payoff matrix obtained from the last iteration of SGInfiniteVI algorithm.

State number	s_0	s_1	s_2	s_3
Spatial representation				

Corresponding matrix — s_0:

Player 1 \ Player 2	L	D	U	R	N
L	3.8;3.9	1.1;1.1	3.8;3.9	3.8;3.9	3.8;3.9
D	3.8;3.9	3.8;3.9	-196;-191	12.8;13.1	3.8;3.9
U	3.8;3.9	3.8;3.9	3.8;3.9	3.8;3.9	-196;-191
R	3.8;3.9	3.8;3.9	3.8;3.9	12.8;13.1	12.8;13.1
N	3.8;3.9	-198;-193	3.8;3.9	3.8;3.9	12.8;13.1

s_1:

Player 1 \ Player 2	L	D	U	R	N
L	3.8;3.9	12.8;13.1	12.8;-181	3.8;3.9	12.8;13.1
D	-196;-191	12.8;13.1	3.8;-191	3.8;3.9	3.8;3.9
U	3.8;3.9	12.8;13.1	3.8;-191	3.8;3.9	3.8;3.9
R	12.8;13.1	42.8;43.7	-187;-181	12.8;13.1	12.8;13.1
N	3.8;3.9	12.8;13.1	12.8;-181	3.8;3.9	12.8;13.1

s_2:

Player 1 \ Player 2	L	D	U	R	N
L	12.8;13.1	12.8;13.1	12.8;13.1	12.8;13.1	12.8;13.1
D	42.8;43.7	42.8;43.7	-157;-151	42.8;43.7	42.8;43.7
U	142;145	142;145	142;145	142;145	-157;-151
R	12.8;13.1	12.8;13.1	12.8;13.1	12.8;13.1	12.8;13.1
N	42.8;43.7	-157;-151	42.8;43.7	42.8;43.7	42.8;43.7

Actions	Down ; Right	Right ; Down	Up ; Down	

Table I
AN EXAMPLE OF POLICY ACTIONS IN A TOROIDAL WORLD.

D. Nash selection functions

In this part we discuss the experimental results of the SGInfiniteVI algorithm and we will see empirically what versions of the function f that will guarantee good results (in case of multiple Nash equilibria). Thus, the tests have been done on 200 policies[6]. For each of these policies, $100,000$ tests were performed[7], and so we get 20 millions tests by experimentation. The experiments were arbitrarily performed with the following scenario: two agents and four objects. The notations of legend indicate different implemented versions of the selection function f, each agent:

- NashMaxTot: selects the first Nash equilibrium that maximizes the payoffs of all agents.
- NashMaxSub: selects the first Nash equilibrium that maximizes its own payoff.
- NashPareto: selects a non-Pareto-dominated Nash equilibrium.

Once the entire actions plan calculated, we wanted to count the percentage of Nash equilibria that are different from one agent to another. This measure will allow us to have a first point of view on the selection methods mentioned previously. Table II shows that in general the percentage of different Nash equilibria remains low and does not exceed (in the worst case) the threshold of 5 percent. It is also clear that the functions NashMaxTot and NashMaxSub are almost equal and offer less different equilibria compared to the function NashPareto.

However, the percentage could not inform us about the quality of solution that can be obtained. Indeed, two different Nash equilibria do not necessarily suggest conflicting actions and therefore we should consider other benchmarks.

[6]The positions of objects are selected randomly following an uniform distribution on the interval $[0, n^2[$.

[7]The positions of agents are selected randomly following an uniform distribution on the interval $[0, n^2[$.

	NashMaxTot	NashMaxSub	NashPareto
4 x 4	2.36	2.46	4.59
5 x 5	0.64	0.61	1.05
6 x 6	0.61	0.59	1.17
7 x 7	1.27	1.38	2.07
8 x 8	0.86	0.83	2
9 x 9	1.19	1.13	1.96
10 x 10	1.55	1.45	1.93
11 x 11	1.08	1.76	2.99

Table II
THE PERCENTAGE OF DIFFERENT NASH EQUILIBRIA ACCORDING TO SEVERAL SELECTION FUNCTIONS.

Thus, some elements of comparison are considered:

- Average number of conflicts per simulation: for this application, no binding hypothesis has been imposed. In particular, the space conflicts are not prohibited. For example, it is quite possible that an agent carrying an object decides to move to an adjacent cell containing another object.
- Average number of livelocks per simulation: the livelock is an endless cycle that prevents agents from reaching a goal state. If agents fall into such cycle, the game never progresses.

It is easy to see that when using the functions NashMaxTot and NashMaxSub give best results: the number of conflicts (figure 4) and livelocks (figure 5) tends towards zero. It seems that these functions lead in most cases agents to the same equilibrium, and otherwise to non-conflicting actions. We also noticed that when agents choose their equilibria by using the function NashPareto then there is a worst number of conflicts and livelocks.

E. Assumption of non-cooperation

Let us remind that the given above results imply a strong assumption: the agents must respect imperatively their plans during the execution phase. Thus in order to check the

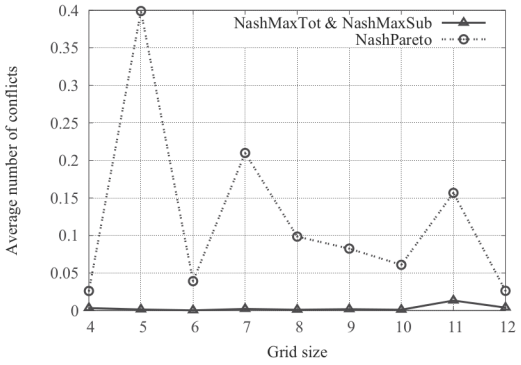


Figure 4. Comparison between Nash selection functions according to the average number of conflicts, with two agents and four objects.

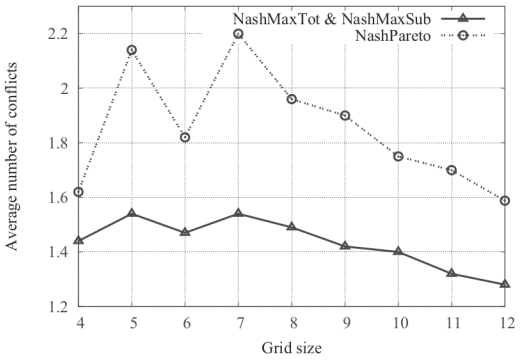


Figure 6. Comparison between Nash selection functions according to the average number of conflicts, with two agents and four objects when the second agent's behavior is randomized.

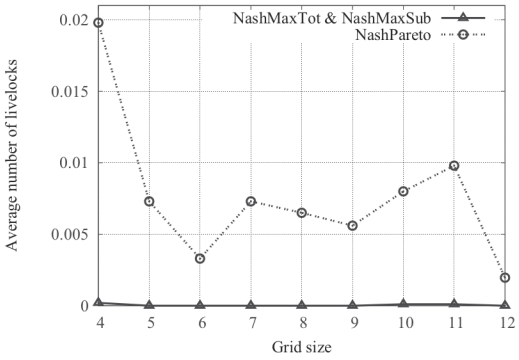


Figure 5. Comparison between Nash selection functions according to the average number of livelocks, with two agents and four objects.

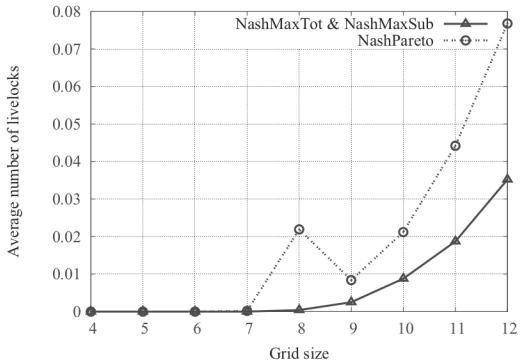


Figure 7. Comparison between Nash selection functions according to the average number of livelocks, with two agents and four objects when the second agent's behavior is randomized.

robustness of the Nash equilibrium solution some complementary experiments are done on less restrictive hypothesis. Indeed, an agent may encounter situations where the other (robot, human, etc) do not cooperate. So we wanted to study the extreme case and assume that one of the two agents has a completely random behavior[8]. These experiments will further point out the Nash selection function that allows (at best) the agent to adapt with such difficult situations.

The small number of livelocks (figure 7) shows that the robot undergoing disruption still manages to accomplish its goal in most tests. Meanwhile, the number of conflicts (figure 6) is also relatively small, for example it is about 1.4 (on average) by 32 steps for a 4×4 grid. The functions NashMaxTot and NashMaxSub confirm previous conclusions by being better than the function NashPareto. All these observations suggest that a solution based on the Nash equilibrium is robust to situations of non-cooperation or failure of a robot and therefore able to cope with much more realistic interactions.

[8]with the same probability distribution over all its actions.

F. Experiments with three agents

We also performed some experiments with a scenario including three agents and three objects, but we found that even with a large battery of tests, the results showed no conflicts or livelocks. We believe that this phenomenon is due to the fact that the number of steps (on average seven steps to move all objects in a grid of 7×7) was not enough for them to have sufficient interaction. Let us note also that the transition to a higher number of agents (> 3) is possible, but only by using a "near Nash equilibrium" [23]. For these reasons we introduced only the results corresponding to the scenario of two agents.

V. Conclusion

In order to coordinate a multi-agent system, the aim of this preliminary work is not only to study the model of general-sum stochastic games but also to develop a planning algorithm based on the dynamic programming and the Nash equilibrium. According to the state of art, this problem is considered as difficult in terms of resolution. Our attempt for

understanding this difficulty consists in the implementation, the evaluation and the validation on an example issued from a classical problem. Different Nash selection functions have been proposed and we have shown experimentally which is the most suitable to have better interactions between agents.

Additionally, our approach still has significant constraints on the environment. Two limitations maybe easily identified:

1) Known games: the algorithm needs to know the payoffs of each agent.
2) Full observability: agents need perfect observations of their environment.

Finally, we find it interesting that the algorithm converged consistently and quickly despite the fact that it is not guaranteed to do so. Hopefully the results of this paper could lead to the investigation of more general settings under which these convergence properties can be proven.

VI. Acknowledgements

This research work was financed by the european community, la Delegation Regionale la Recherche et la Technologie, le Ministere de l'Education Nationale, de la Recherche et de la Technologie, la region Nord Pas de Calais, le Centre National de la Recherche Scientifique, et l'Agence Nationale de la Recherche: projet AMORCES.

References

[1] A. Beynier and A.-I. Mouaddib, "An iterative algorithm for solving constrained decentralized markov decision processes," in *AAAI'06: proc. of the 21st national conf. on Artificial intelligence*. AAAI Press, 2006, pp. 1089–1094.

[2] R. Aras, A. Dutech, and F. Charpillet, "Cooperation in stochastic games through communication," in *AAMAS '05: Proc. of the fourth int. joint conf. on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2005, pp. 1197–1198.

[3] S. Ganzfried and T. Sandholm, "Computing equilibria in multiplayer stochastic games of imperfect information," in *IJCAI'09: Proc. of the 21st int. jont conf. on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 140–146.

[4] M. Kearns, Y. Mansour, and S. Singh, "Fast planning in stochastic games," in *In Proc. UAI-2000*. Morgan Kaufmann, 2000, pp. 309–316.

[5] A. Burkov and B. Chaib-draa, "Distributed planning in stochastic games with communication," *Machine Learning and Applications, Fourth Int. Conf. on*, pp. 396–401, 2008.

[6] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[7] S. Zilberstein, R. Washington, D. Bernstein, and A. Mouaddib, "Decision-theoretic control of planetary rovers," in *Revised Papers from the Int. Seminar on Advances in Plan-Based Control of Robotic Agents*. London, UK: Springer-Verlag, 2002, pp. 270–289.

[8] J. V. Neumann and O. Morgenstern, *Theory of games and economic behavior*. Princeton University Press, Princeton, 1944, second edition in 1947, third in 1954.

[9] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. The MIT Press, July 1994.

[10] L. Shapley, "Stochastic games," *Proc. of the National Academy of Sciences USA*, pp. 1095–1100, 1953.

[11] J. F. Nash, "Equilibrium points in n-person games," *Proc. of the National Academy of Sciences of the United States of America*, vol. 36, no. 1, pp. 48–49, January 1950.

[12] A. Clodic, R. Alami, V. Montreuil, S. Li, B. Wrede, and A. Swadzba, "A study of interaction between dialog and decision for human-robot collaborative task achievement," in *Int. Symp. on Robot and Human Interactive Communication*. Jeju Island, Korea: IEEE, 2007.

[13] F. Dehais, S. Mercier, and C. Tessier, "Conflicts in human operator — unmanned vehicles interactions," in *EPCE '09: Proc. of the 8th Int. Conf. on Engineering Psychology and Cognitive Ergonomics*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 498–507.

[14] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," Stanford University, Tech. Rep., 2003.

[15] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[16] V. Conitzer and T. Sandholm, "New complexity results about nash equilibria," *Games and Economic Behavior*, vol. 63, no. 2, pp. 621–641, July 2008.

[17] F. Fischer, M. Holzer, and S. Katzenbeisser, "The influence of neighbourhood and choice on the complexity of finding pure nash equilibria," *Inf. Process. Lett.*, vol. 99, no. 6, pp. 239–245, 2006.

[18] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *In Proc. of the Eleventh Int. Conf. on Machine Learning*. Morgan Kaufmann, 1994, pp. 157–163.

[19] J. Hu and M. Wellman, "Nash q–learning for general–sum stochastic games," *Journal of Machine Learning Research*, vol. 4, pp. 1039–1069, 2003.

[20] M. Zinkevich, A. Greenwald, and M. Littman, "Cyclic equilibria in markov games," in *NIPS*, 2005.

[21] Y. Vorobeychik and M. Wellman, "Stochastic search methods for nash equilibrium approximation in simulation-based games," in *Proc. AAMAS '08*, Richland, SC, 2008, pp. 1055–1062.

[22] R. Aras, A. Dutech, and F. Charpillet, "Computing the equilibria of bimatrix games using dominance heuristics," *Tools with Artificial Intelligence, IEEE Int. Conf.*, pp. 773–782, 2006.

[23] C. Daskalakis and A. M. C. Papadimitriou, "A note on approximate nash equilibria," *Theor. Comput. Sci.*, vol. 410, no. 17, pp. 1581–1588, 2009.