GPU-based CSP for Action Planning

Stéphane Cardon
CREC Saint-Cyr

MOTIVATIONS

- F.E.A.R.[1] : First GAME using planning
  - Better feeling of realism
- Now Shadow of Mordor (same game development studio) :
  - 50 Non-Playable Characters at same time
- Is-it possible to plan 100 or 1000 NPCs ?
  - Military simulation, Wargame
  - Strong constraint : compute solution in one frame (16 ms)

1. ORKIN, J. 2006. Three states and a plan : The A.I. of F.E.A.R.

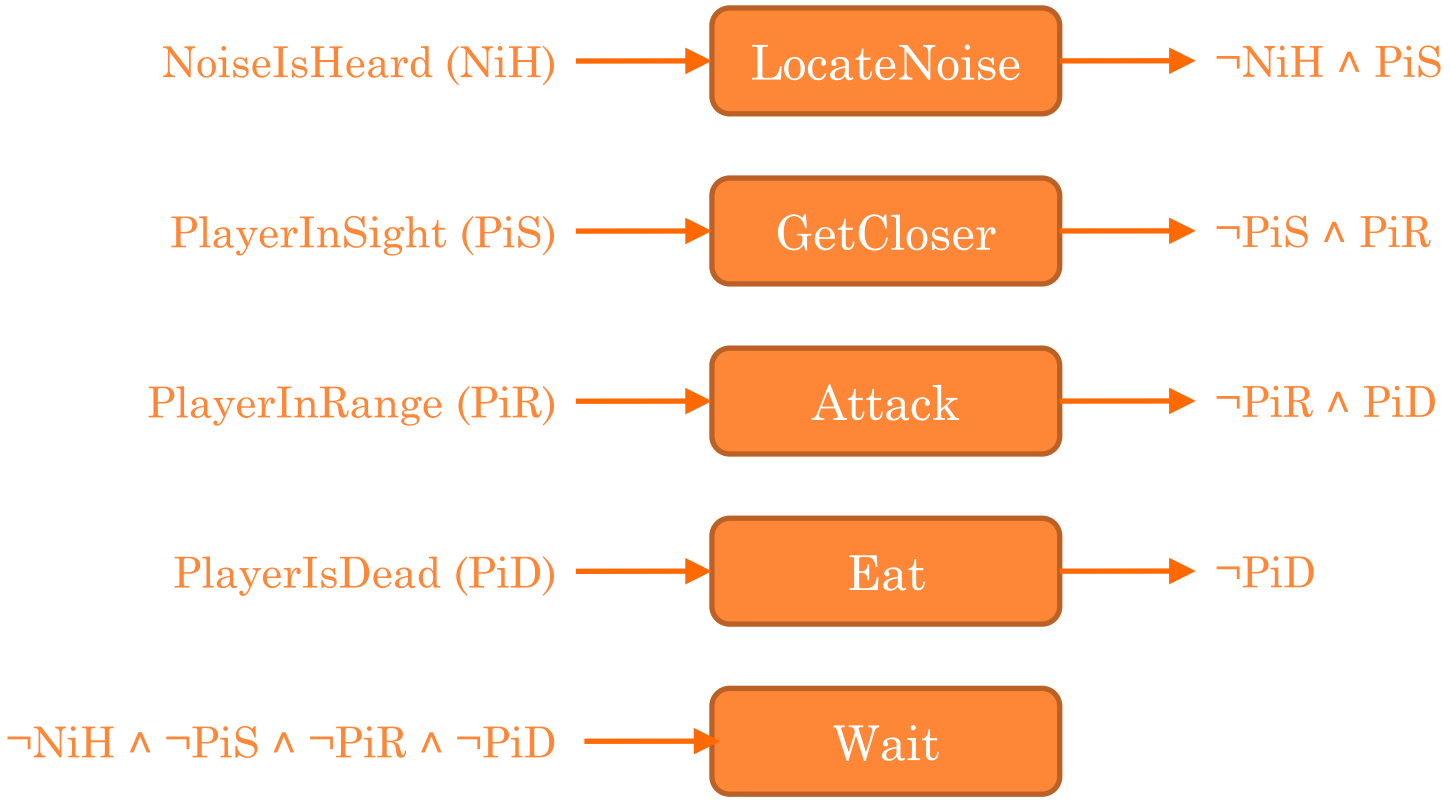
Plan

1. Planning in games
2. Binary CSP for planning
3. Solving binary CSP using GPU
4. Conclusion

Zombie example

- One goal : eating
- Possible actions to reach this goal :
  - Wait, Locate noise, Get closer, Attack, Eat
- Effects, for planning, are supposed to be certain :
  - After attacking, player will be dead
  - In fact, player may be dead

ACTIONS OF A ZOMBIE



Planning in games

- Finding a sequence of actions to reach the goal from initial state
- Uncertainty on effects :
  - Plans of short size[2]
  - Re-plans regularly
- Decouple game world state to decision state :
  - Player and Zombie positions/Player is in sight

2. JACOPIN, É. 2014. Game AI planning analytics.

Why CSP ?

- Previous work shows that it is possible to increase resolution of planning problem seen as CSP one[3]

3. VAN BEEK, P., and CHEN, X. 1999. CPlan : A constraint programming approach to planning.

Write CSP from planning problem[4]

- Fix an horizon : the length of the biggest solution
- For a Zombie, size is 4 :
  - Locate noise
  - Get closer
  - Attack
  - Eat

4. POOLE, D. L., and MACKWORTH, A. K. 2010. Artificial Intelligence : Foundations of computational agents.

CSP variables

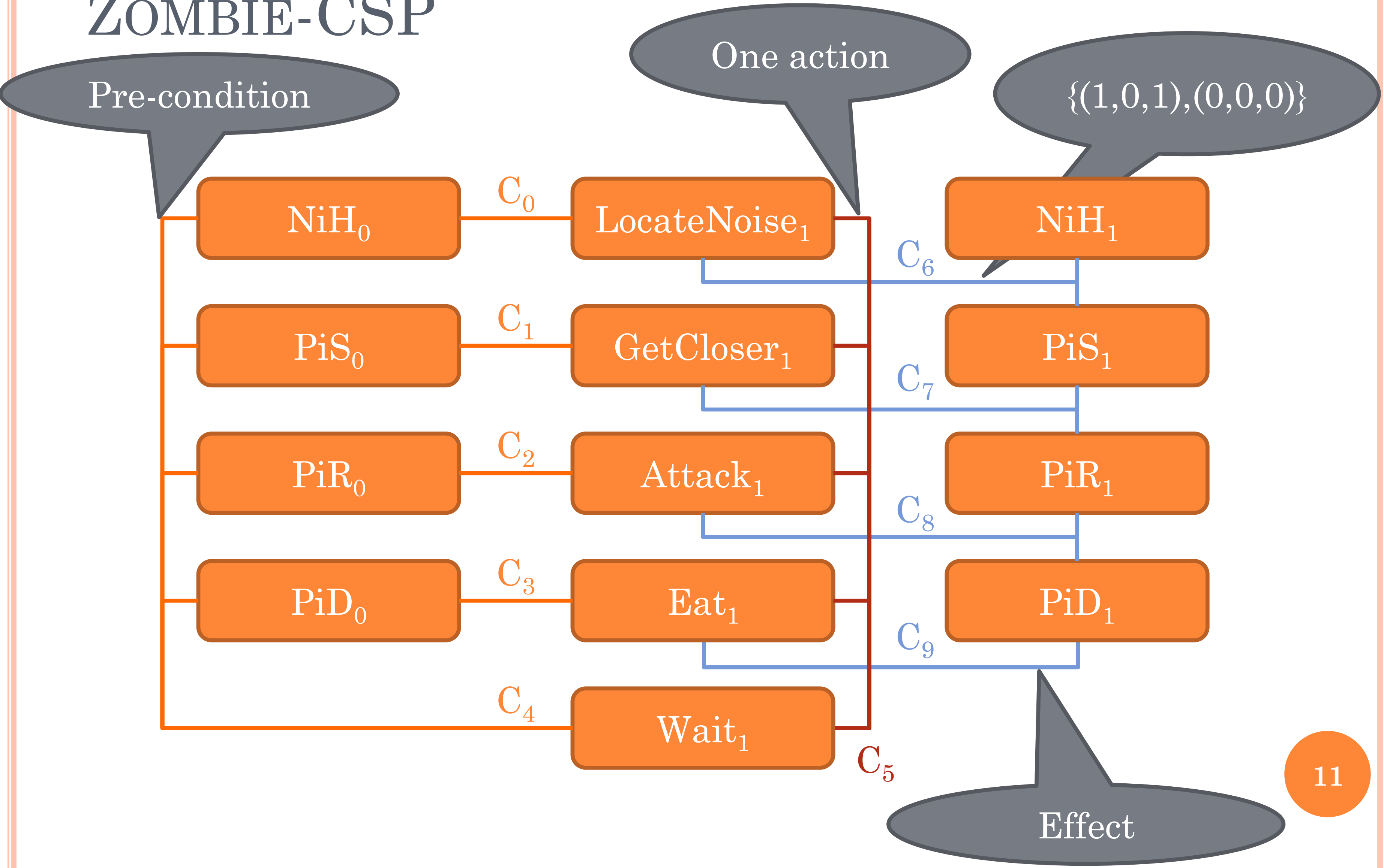
- The action to do at stage t : <u>Attack</u>$_t$
- Variables of planning problem for each stage : <u>Noise is heard</u>$_t$

CSP constraints

- Pre-conditions of planning actions
- Effects of planning actions
- Only one action should be chosen at stage *t*

- Unary constraints to represents initial and goal states

Zombie-CSP

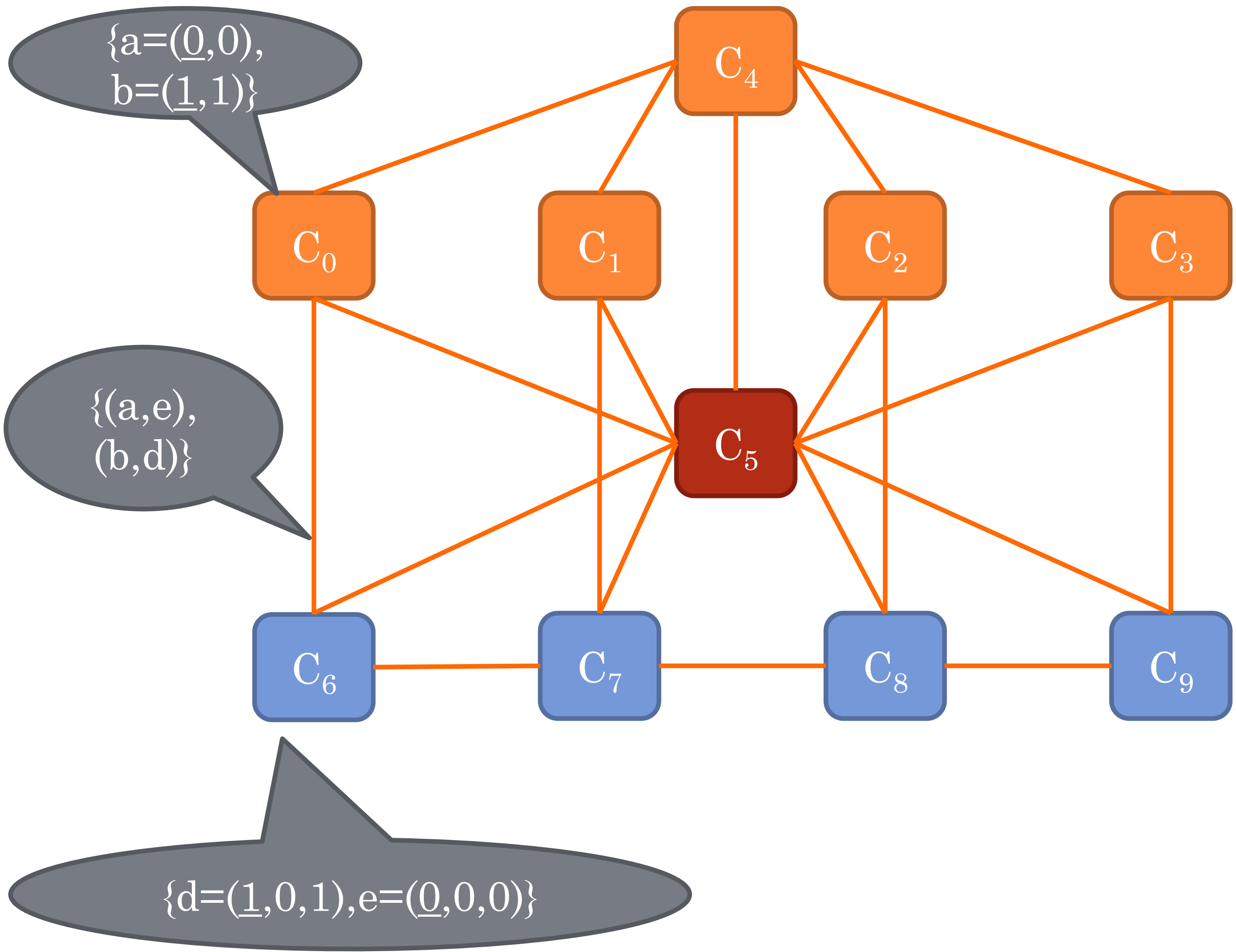

Pre-condition
One action
{(1,0,1),(0,0,0)}
NiH0
PiS0
PiR0
PiD0
C0
C1
C2
C3
C4
LocateNoise1
GetCloser1
Attack1
Eat1
Wait1
C5
C6
C7
C8
C9
NiH1
PiS1
PiR1
PiD1
Effect

Binary CSP

- Any non-binary problem can be transformed in binary ones (e.g. using Dual representation[5]) :
  - $Variable^{Dual}$ = constraint
    - value = a tuple
  - $Constraint^{Dual}$ = binary between two constraints having variables in common
    - tuple = couple of tuples having same values for variables in common

5. F. Bacchus, X. Chen, P. van Beek and T. Walsh; Binary vs. Non-Binary Constraints

Zombie binary CSP



CUDA IN BRIEF

- Based on Simple Instruction Multiple Datas (SIMD) architecture seen as SIMT (Threads)
- A multiprocessor unit (SM or SMX) execute a warp of 32 threads
  - In SMX, hide latency of memory access by calculus
  - Warp access 32 successive elements (a thread 1)
  - Branching must be avoided (unused threads) or reduced to two branches
- Threads are organized in a block and blocks in a grid

GPU-ARC-CONSISTENCY

- Based on AC3
- Launch a grid of N blocks composed of D_{max} AC-threads
- An AC-thread work with $X=a$
  - Consider all variables Y in relation with X
    - Test each value b of domain of Y to find a support
    - If no support is found, mark a as deleted in D_X
- Re-launch grid until no values are deleted

EXPERIMENTS

- On quasi-random, hanoi, graph coloring, n-queens and bandwidth coloring problems
- Speed-up (time of CPU AC3/time of GPU AC) obtained varying between 0.33 to 2.43
- Worst-case : hanoi problem (re-launch grid, up to 60 times)

GPU-Path-Consistency (1/2)

- GPU needs lot of calculus in comparison of loaded datas to be efficient
- Based on PC-2001
- Grid of $\frac{N \times (N-1)}{2}$ blocks of 1024 PC-threads
- A block concern two variables *X* and *Y* (s.t. *X is "defined before" Y*)
- PC-threads must deal with all couples (X=a,Y=b) with coalescent access

GPU-PATH-CONSISTENCY (2/2)

- Consider all dedicated couple (a,b) :
  - For each variable Z in relation with X and Y :
    - Test all possible values c of Z in order to find support for X=a and Y=b
    - If no support found, mark couple (a,b) as deleted in relation between X and Y
- Re-launch grid until no deletion
- Launch GPU-AC

More experiments

- On random problems :
  - 40 variables
  - Domains of maximal size of 25
  - No more than 180 constraints
- Characterized by :
  - Tightness : probability of having two values in relation
  - Density : probability of having a constraint between two variables

RESULTS

Tightness	Density	Speed-up GPU AC	Speed-up GPU PC
0.1	0.96	0.7	**10.23**
0.2	0.53	0.69	**8.66**
0.5	0.23	0.73	**6.06**
0.65	0.17	0.77	**3.44**
0.8	0.13	0.97	**2.1**
0.9	0.1	**2.01**	**1.54**

- Tightness and Density are around 0.5 for Zombie binary CSP

GPU-PC seems to be efficient : Maintaining GPU-PC algorithm ?

Conclusion/Future works

- Even with a speed-up of 2 or more for PC, it is not practical for real-time solving (less than 16 ms) :
  - Pre-resolution phase for modeling ?
  - Other models or modeling ?
- Current work :
  - Grounded-planning on GPU
  - Apply on blocks world (5 blocks, 256 problems in parallel, less or equal than 12 ms)[6]
  - Development : toy game with hundreds of Zombies

6. S. Cardon and É. Jacopin; Poster - Game AI planning : Which GPU for how many NPCs?