Monte-Carlo Fork Search for Cooperative Path-Finding

Bruno Bouzy

Paris Descartes University

Journée du LIPADE

16 juin 2015

Outline

- Cooperative Path-Finding (CPF)
- Monte-Carlo Tree Search (MCTS)
- (Nested) Monte-Carlo Search (MCS)
- Monte-Carlo Fork Search (MCFS)
- Nested MCFS (NMCFS)
- CPF simulations
- Experimental results
- Conclusions

The problem

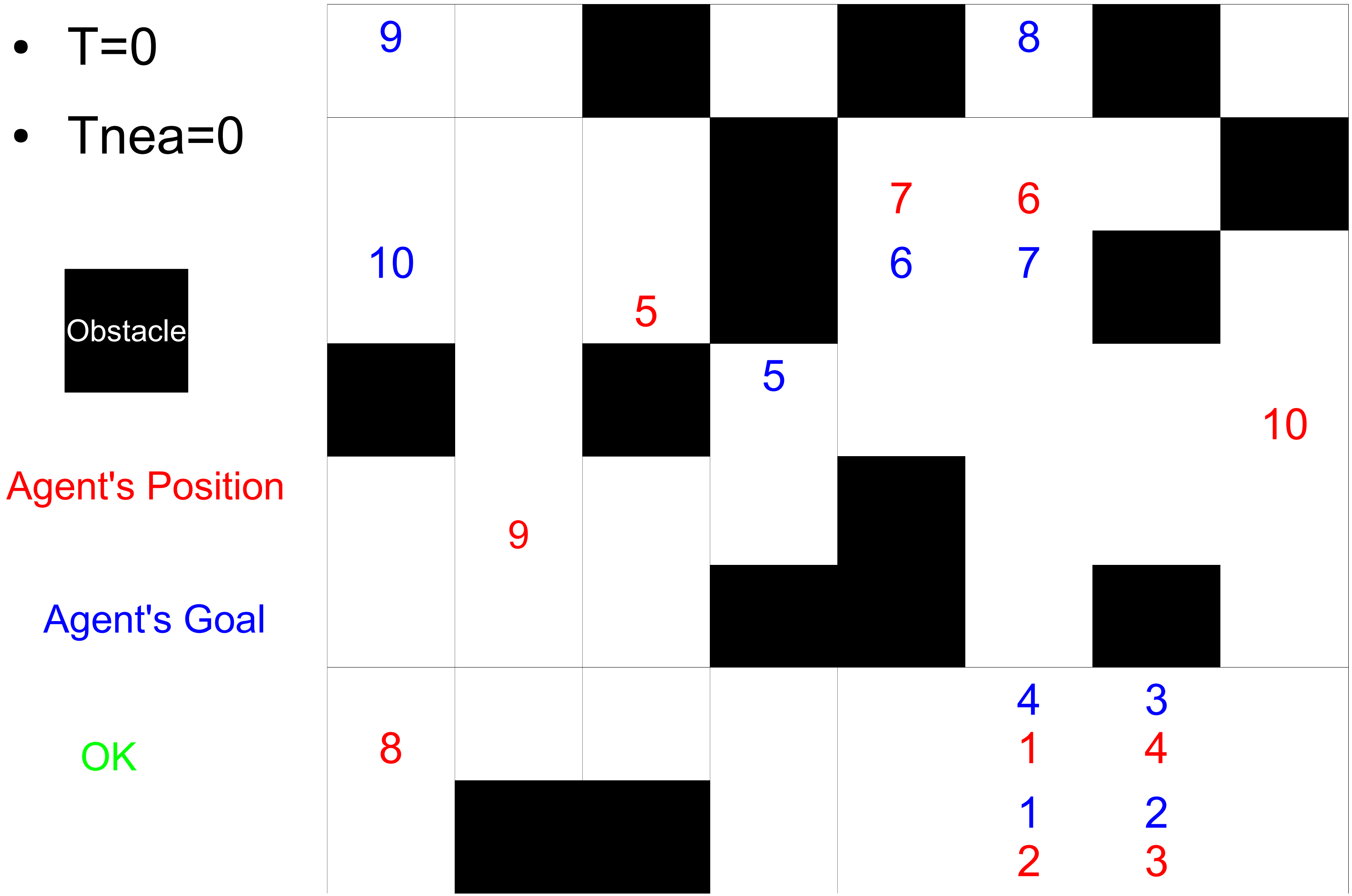
Background

Contribution

The solution

CPF example

- T=0
- Tnea=0



Each number corresponds to an agent.

CPF example

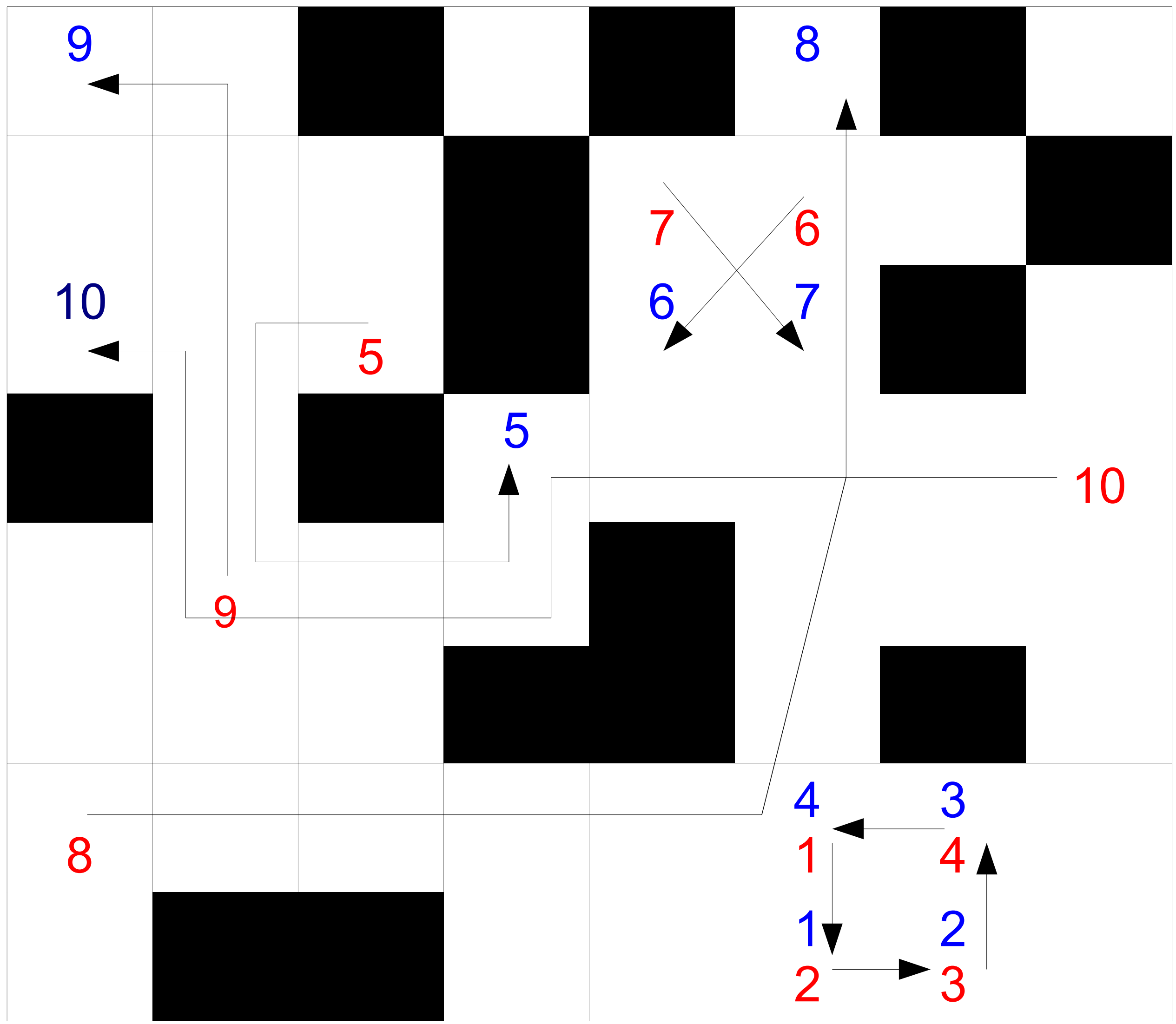
- T=0
- Tnea=0

Obstacle

Position

Goal

OK



Individual Path

CPF rules

- (On a graph)
- On a grid
  - 4-connectivity (or 8-connectivity)
  - obstacles
- Rules:
  - An agent can move on the neighbouring cell or stay
  - Rule 0: no two agents can be on the same cell
  - Rule 1: no two agents can swap
  - Circularity is possible

CPF example

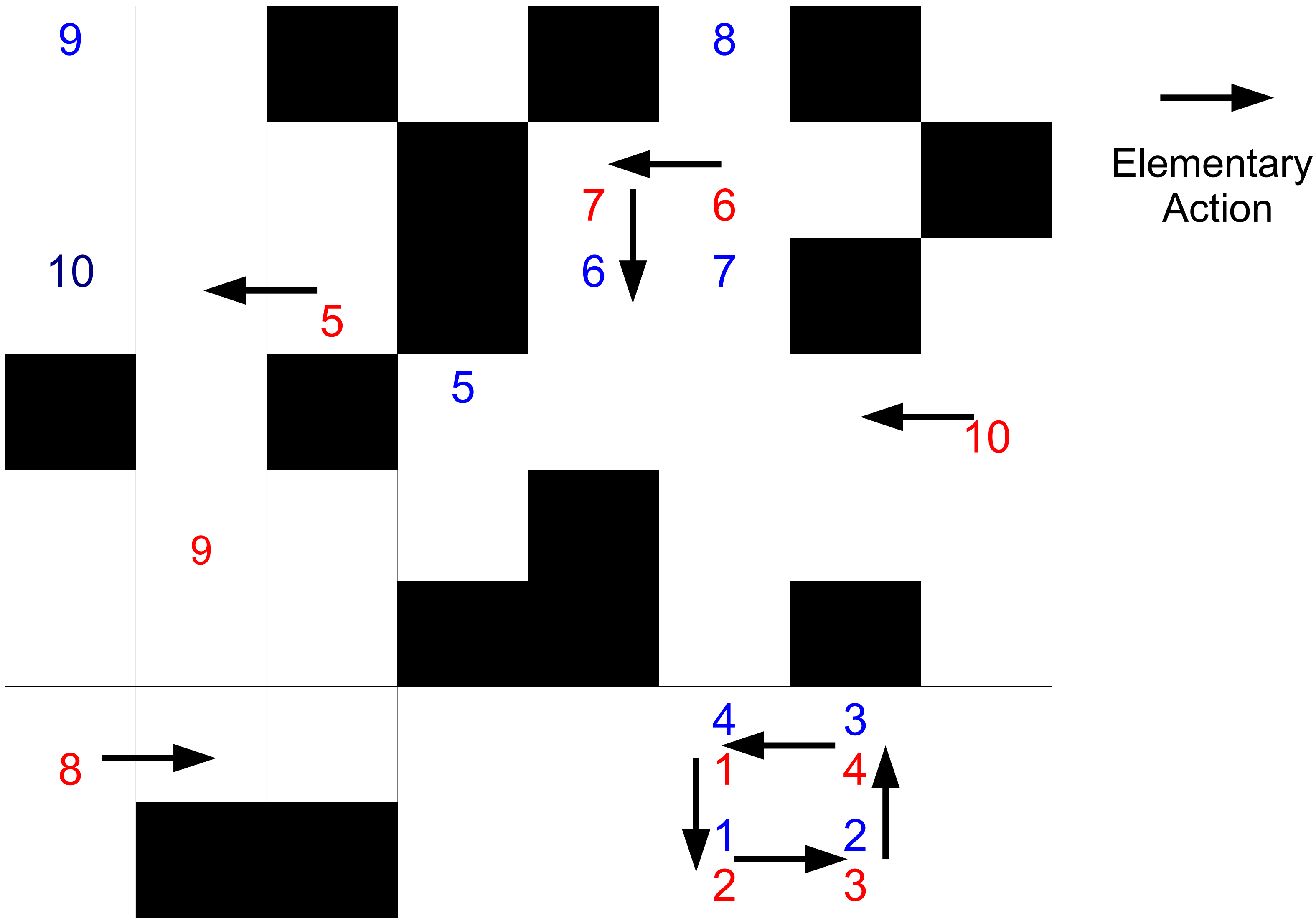
- T=0
- nea=0

Obstacle

Position

Goal

OK



CPF example

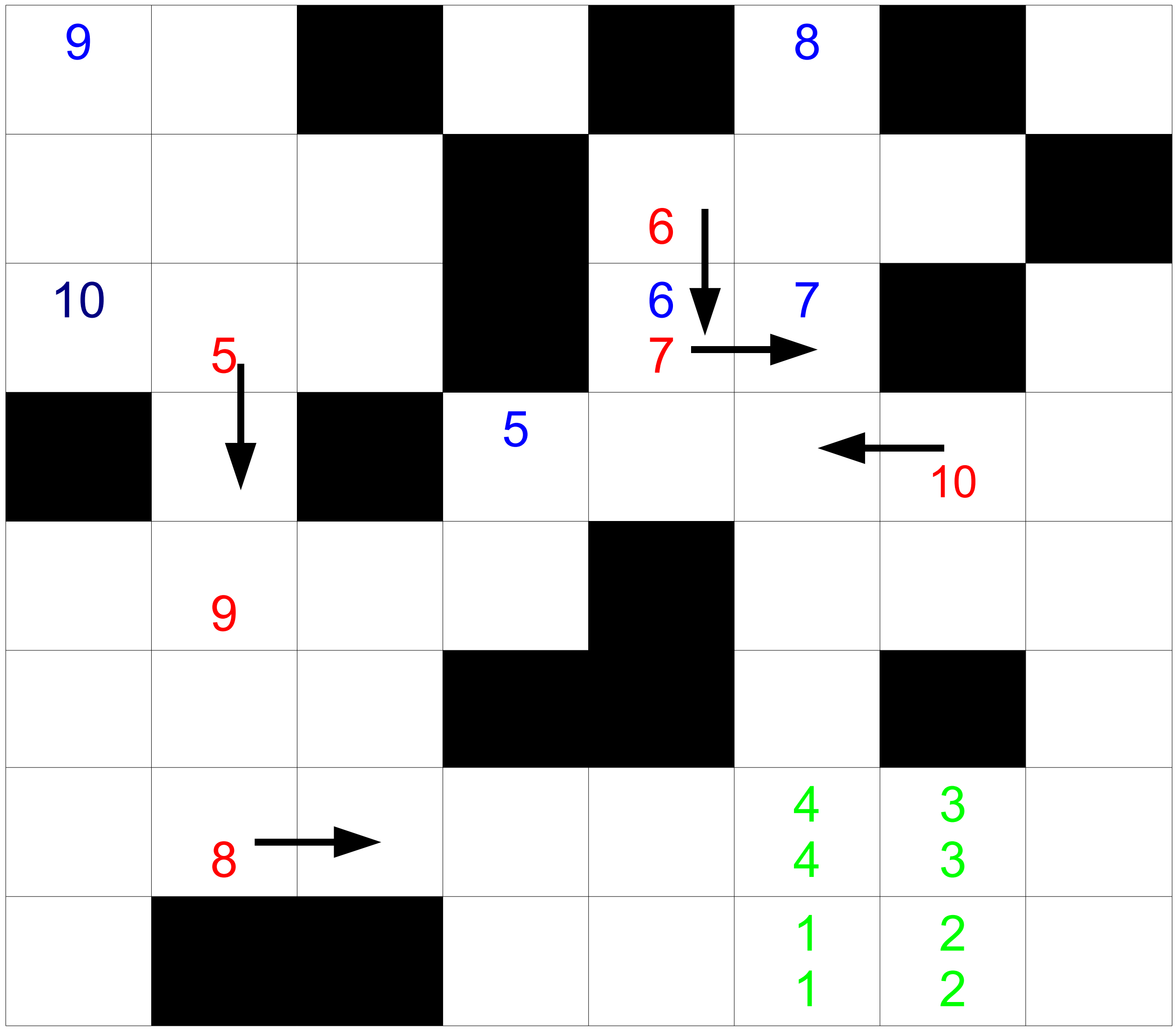
- T=1
- Tnea=9
- nea=5

Obstacle

Position

Goal

OK



Elementary Action

CPF example

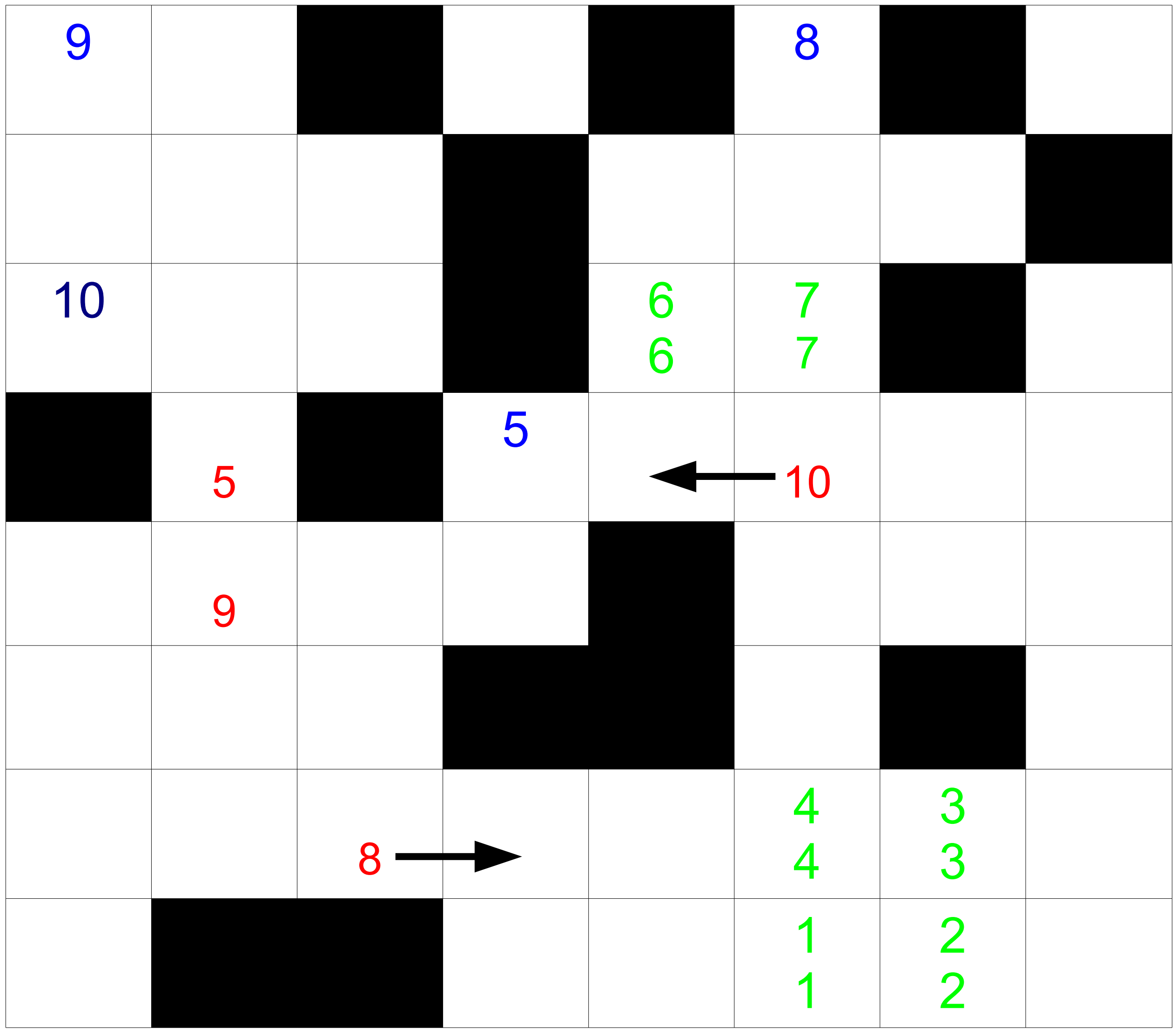
- T=2
- Tnea=14
- nea=2

Obstacle

Position

Goal

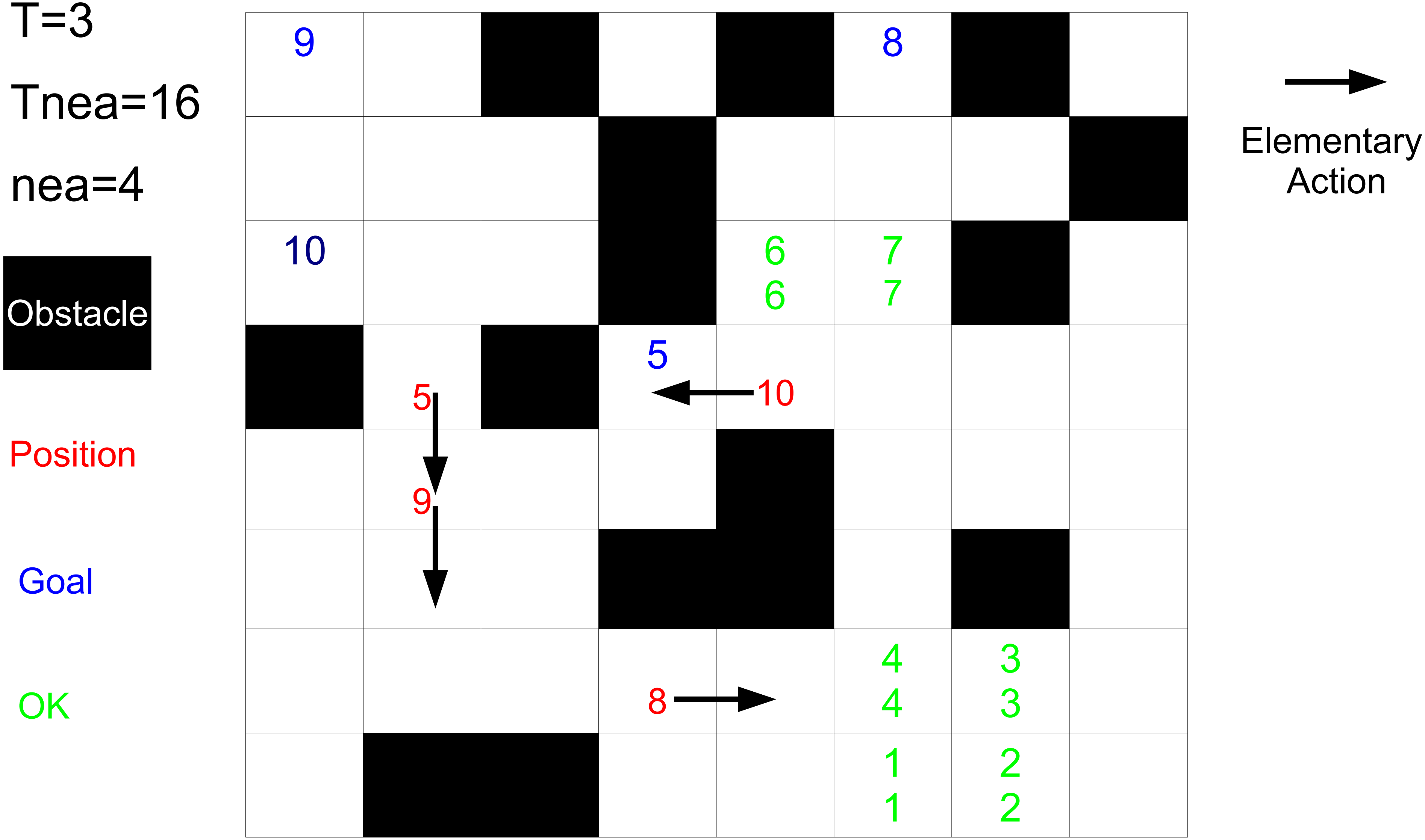
OK



Elementary Action

CPF example

- T=3
- Tnea=16
- nea=4



CPF example

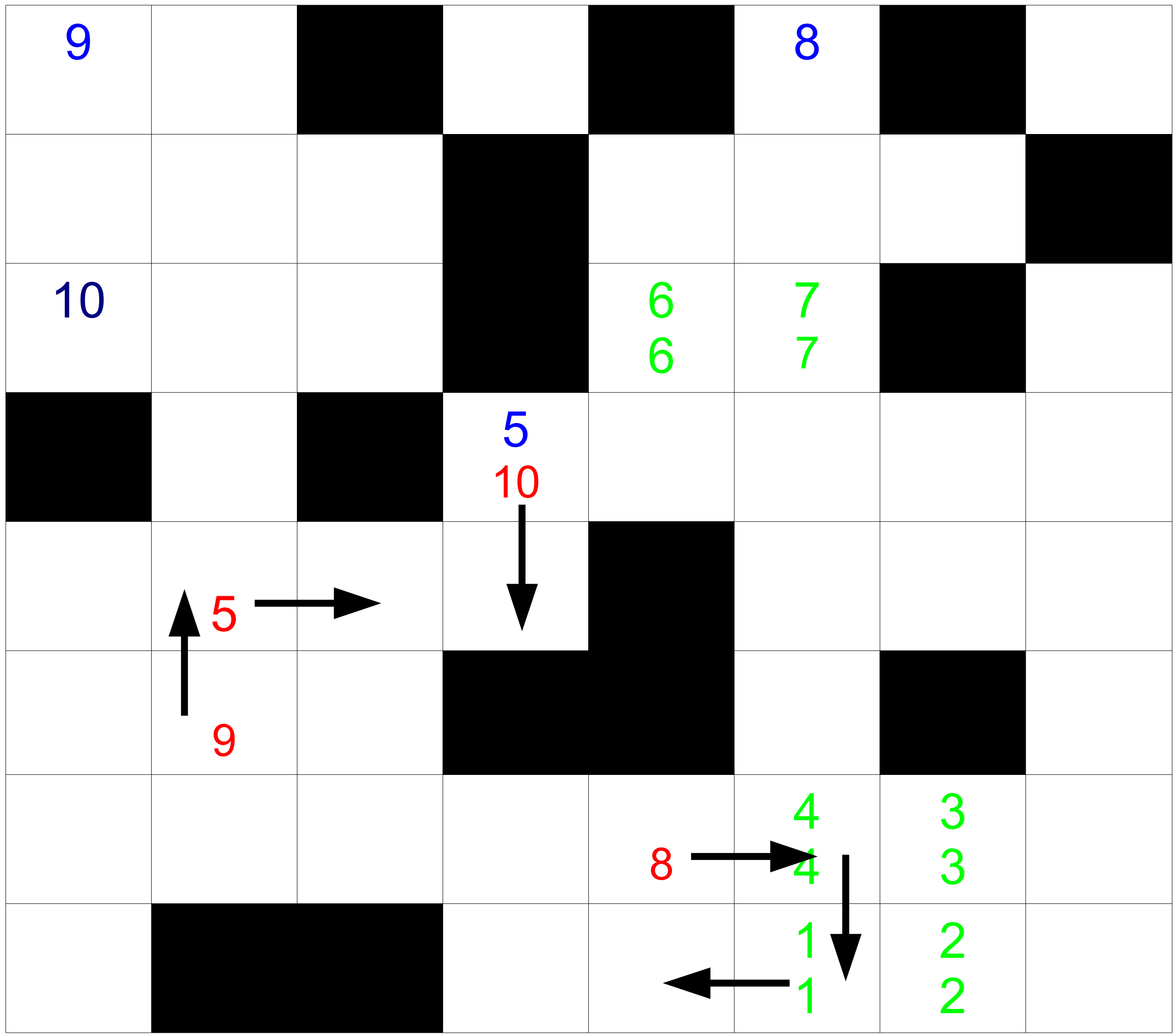
- T=4
- Tnea=20
- nea=6

Obstacle

Position

Goal

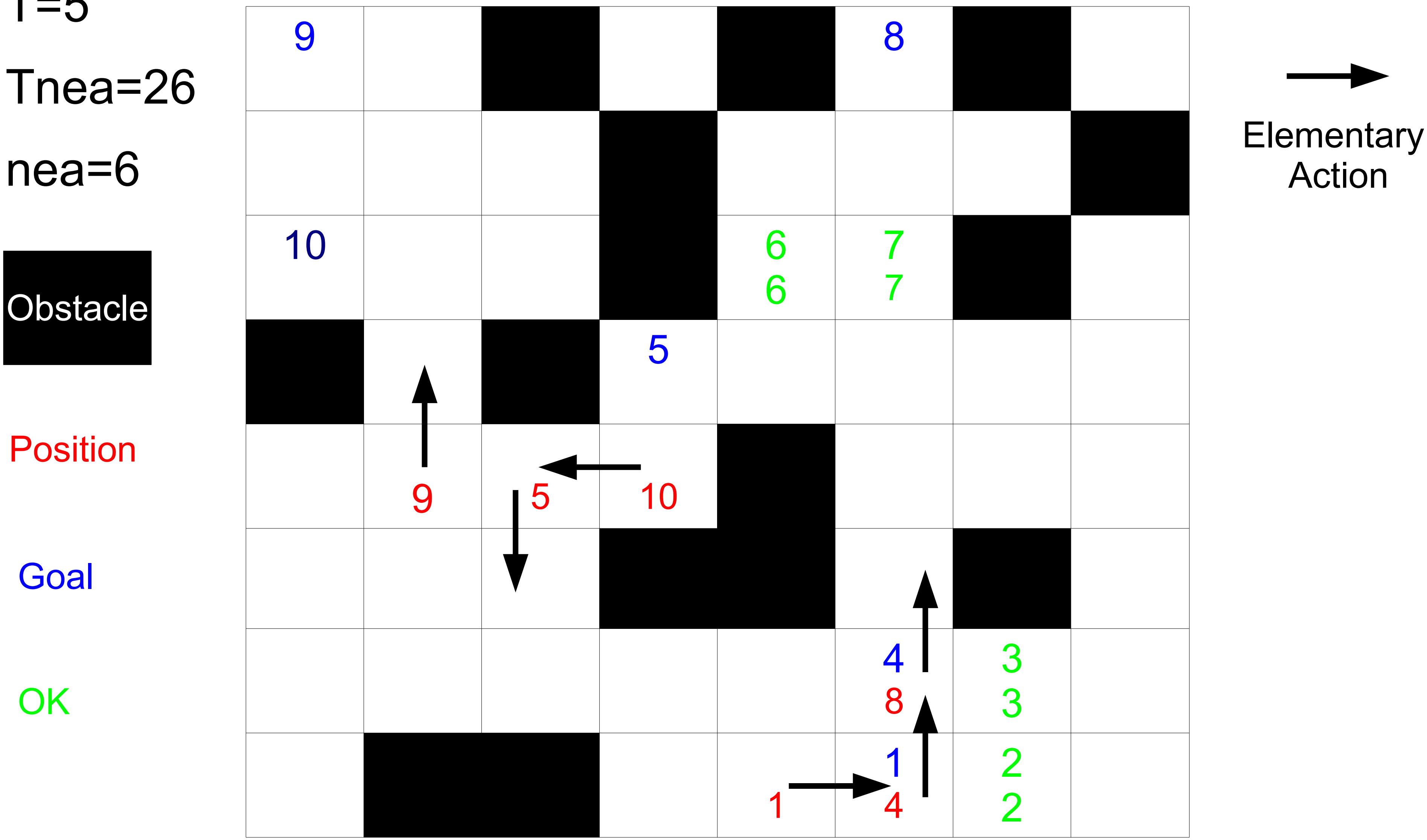
OK



Elementary Action

CPF example

- T=5
- Tnea=26
- nea=6



CPF example

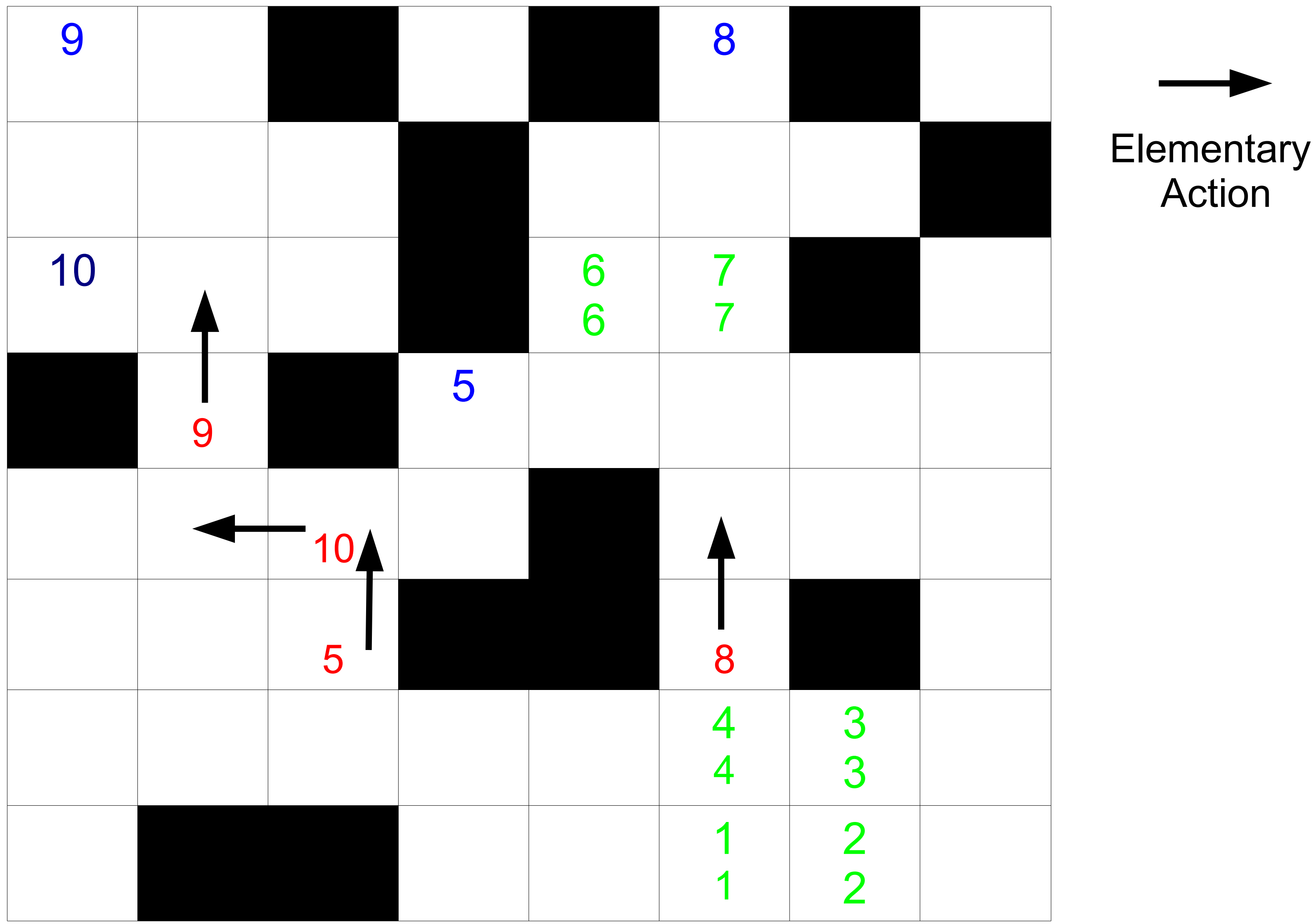
- T=6
- Tnea=32
- nea=4

Obstacle

Position

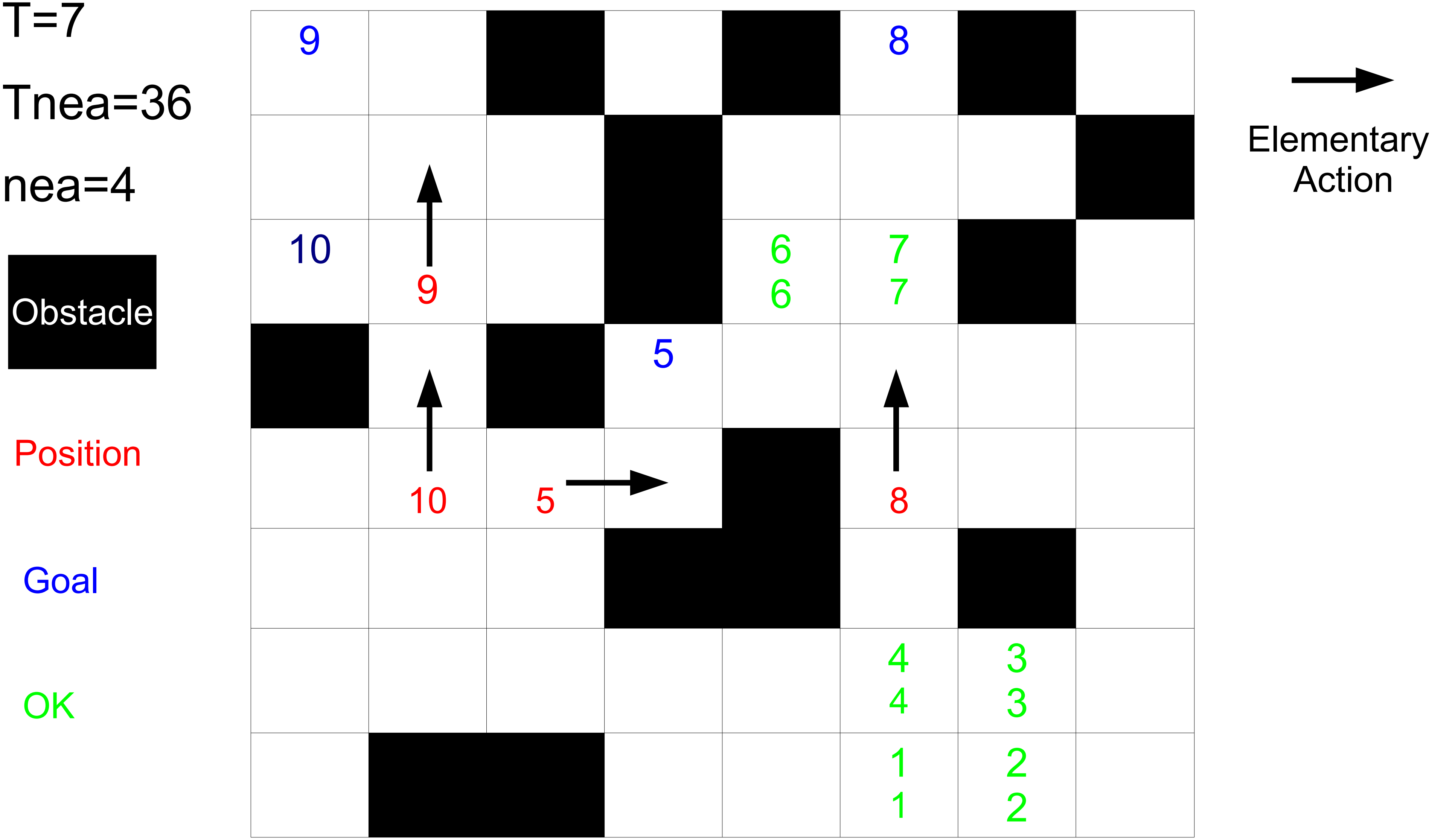
Goal

OK



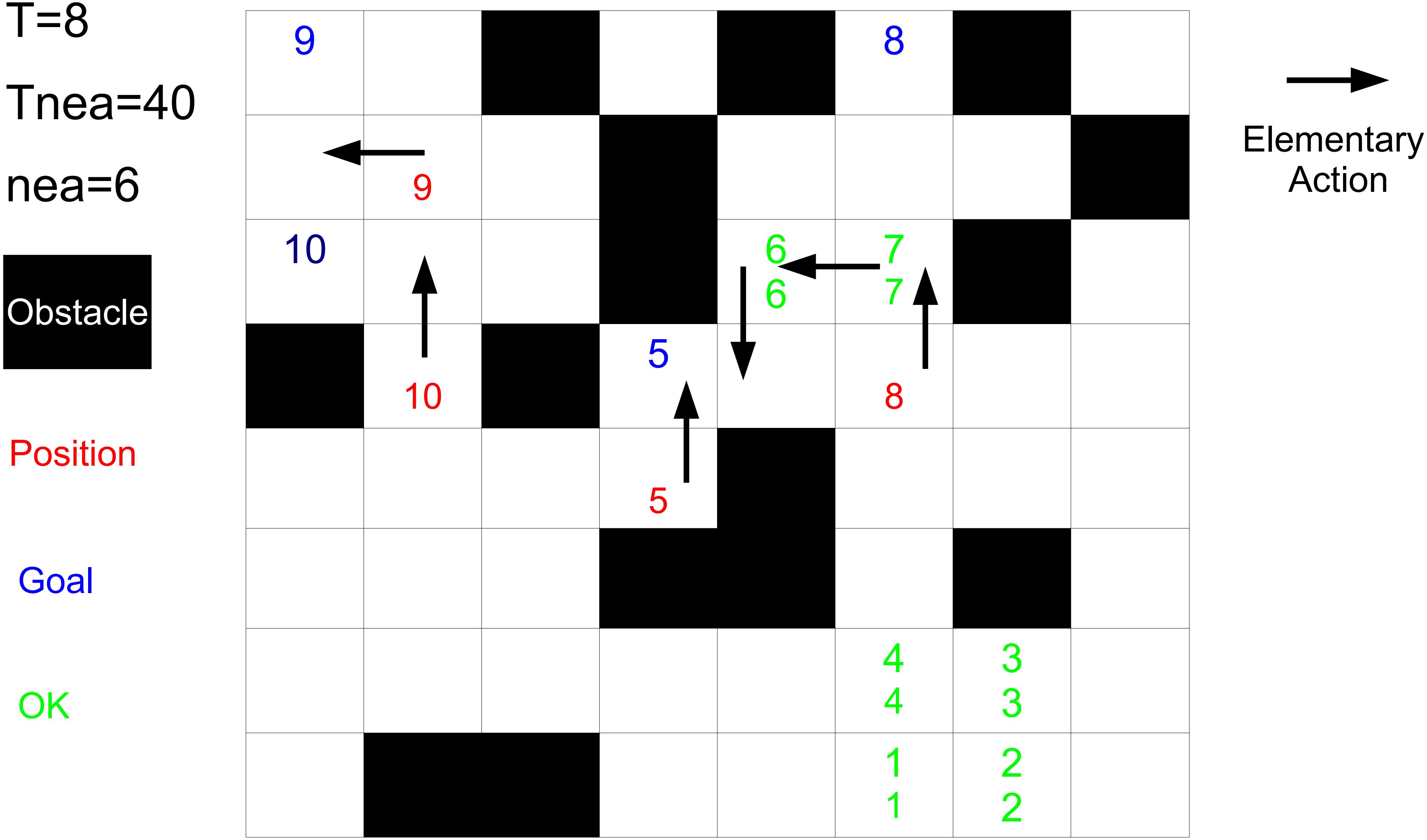
CPF example

- T=7
- Tnea=36
- nea=4



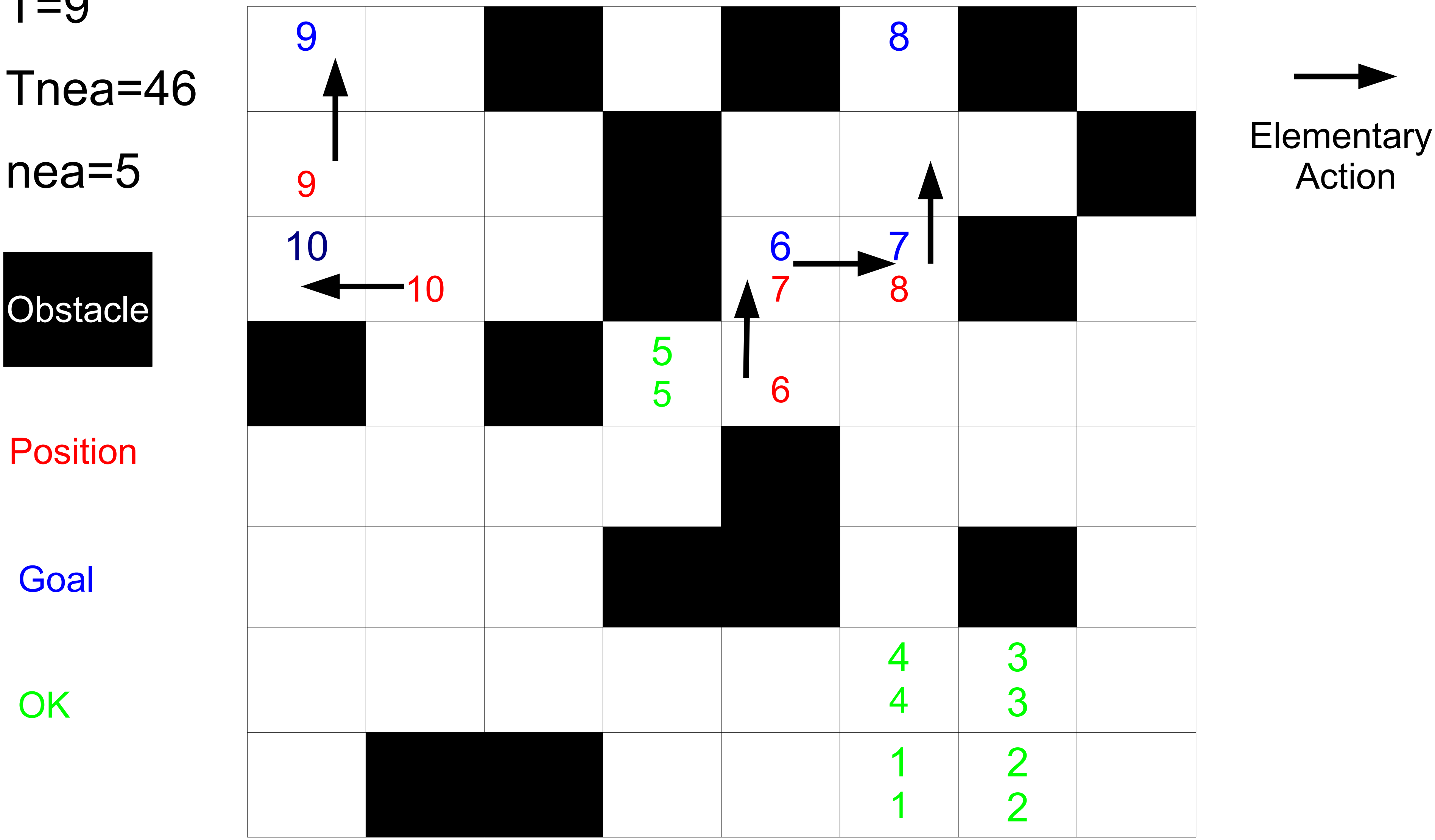
CPF example

- T=8
- Tnea=40
- nea=6



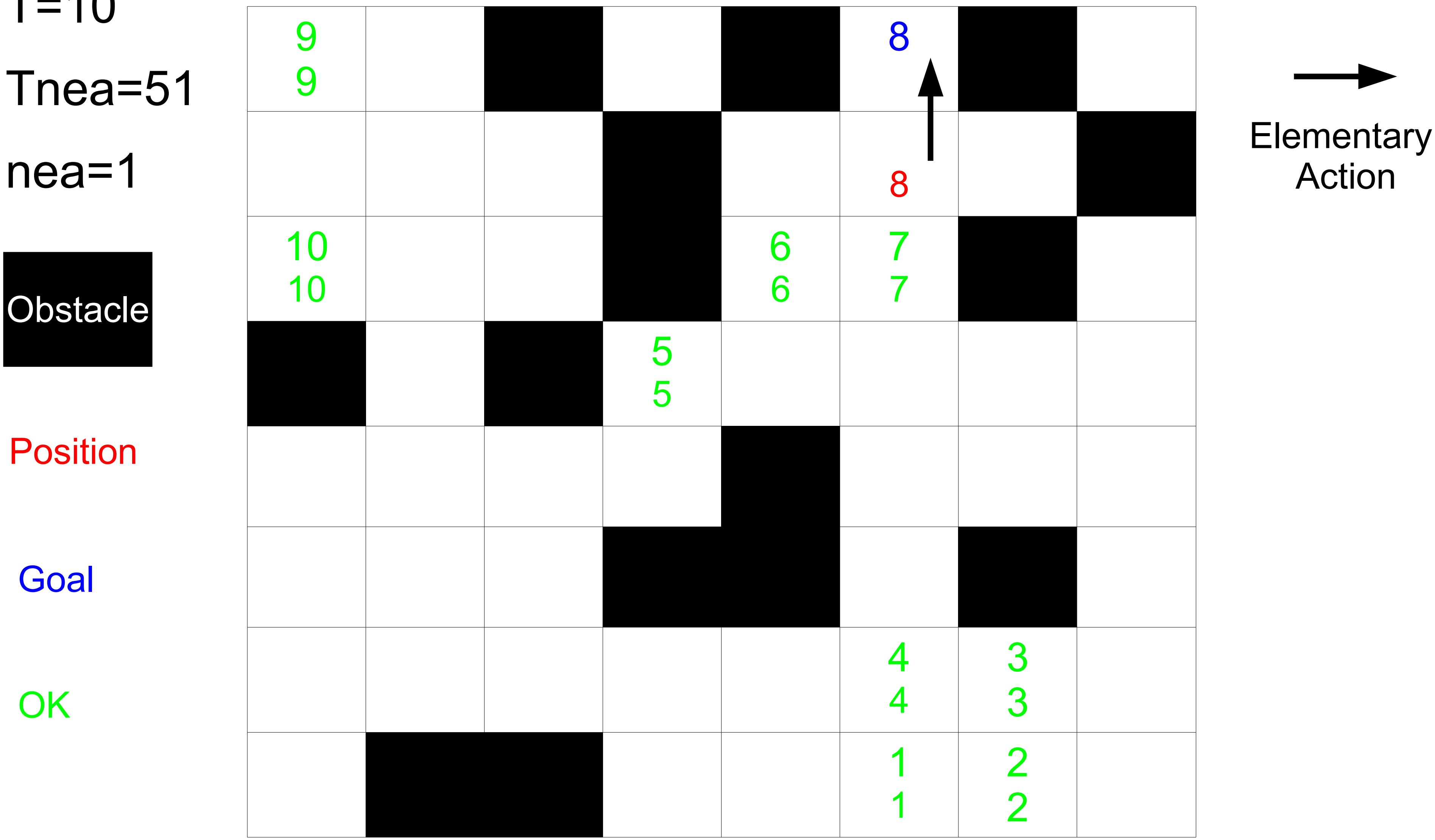
CPF example

- T=9
- Tnea=46
- nea=5



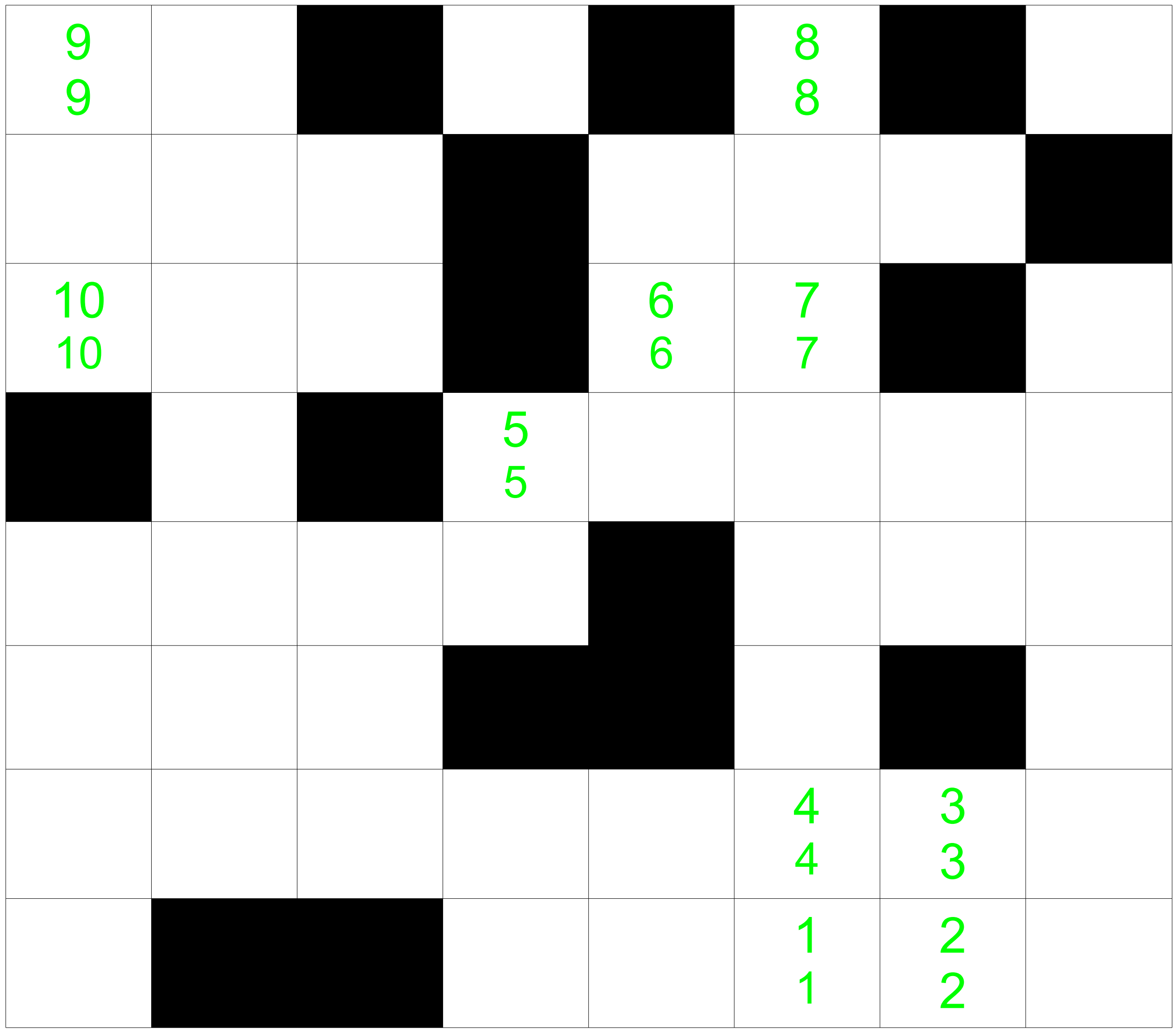
CPF example

- T=10
- Tnea=51
- nea=1



CPF example : end !

- T=11
- Tnea=52



Obstacle

Position

Goal

OK

Elementary Action

CPF optimization target

- Sequentiality
  - sum of individual costs
  - i. e. Total number of elementary actions (Tnea)
  - Most of work in the literature
- Simultaneity
  - Global elapsed time (T) = Number of timesteps
  - Our work + TOMPP (Yu 2012)

Complexity

- Number of joint actions is exponential in the number of agents
  - 10 agents and 5 actions yield $5^{10} \approx 10^7$ joint actions
- « Best first » searches (MCTS, A*) do not work

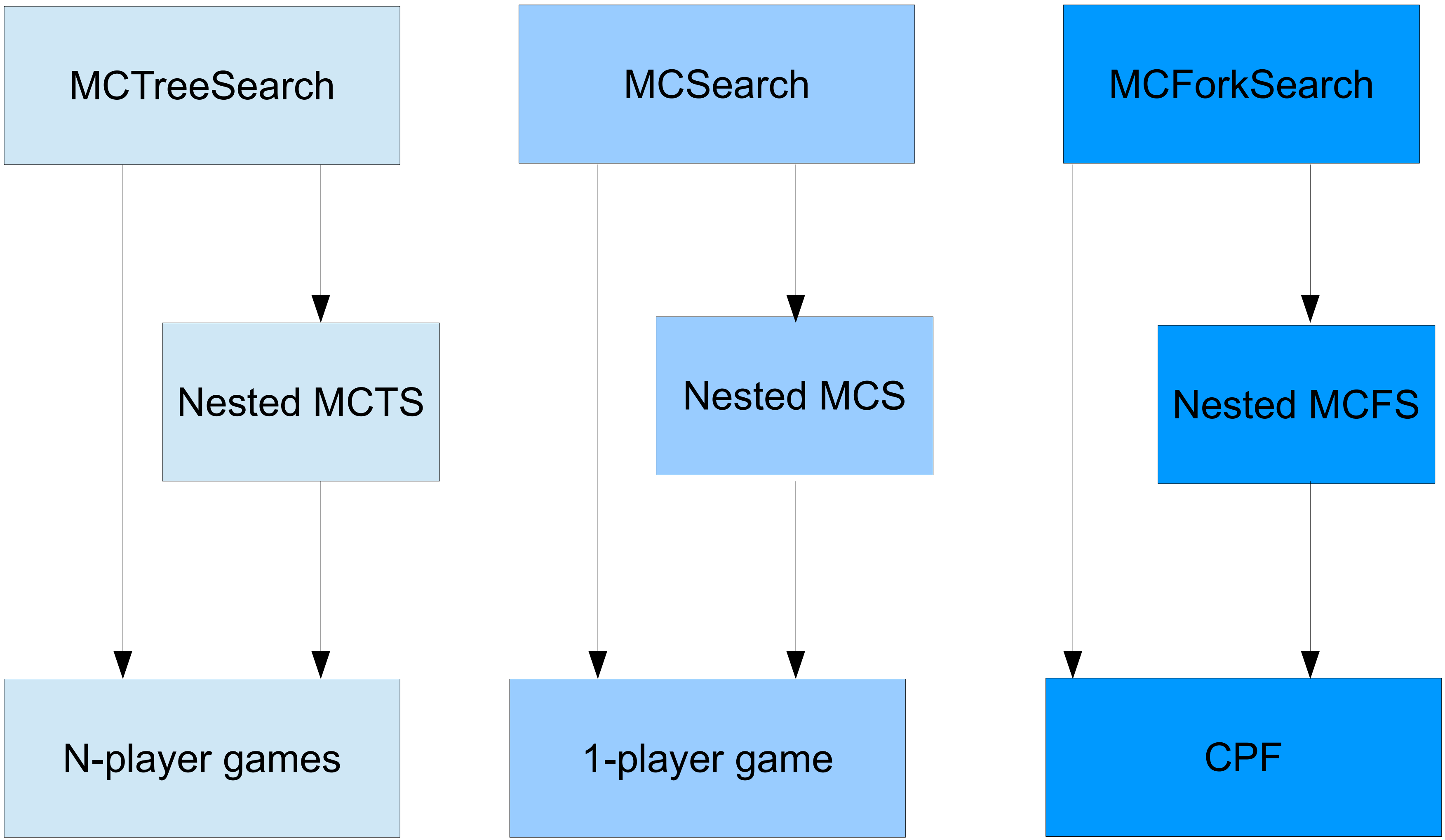
  (even on the simple example)
- Challenging !

CPF previous work

- WHCA* (Silver 2006)
- A* + Operator Decomposition (OD) (Standley 2010)
- A* + Independent Detection (ID) (Standley 2011)
- Incremental Cost Tree Search (ICTS) (Sharon 2011)
- Multi-Agent Path Planning (MAPP) (Botea 2011)
- Push & Swap (Luna 2011)
- TASS (Khorshid 2011)
- Time-Optimal Multi-Agent Path Planning (TOMPP) (Yu 2012)
- CPF video game benchmarks (Sturtevant 2012)

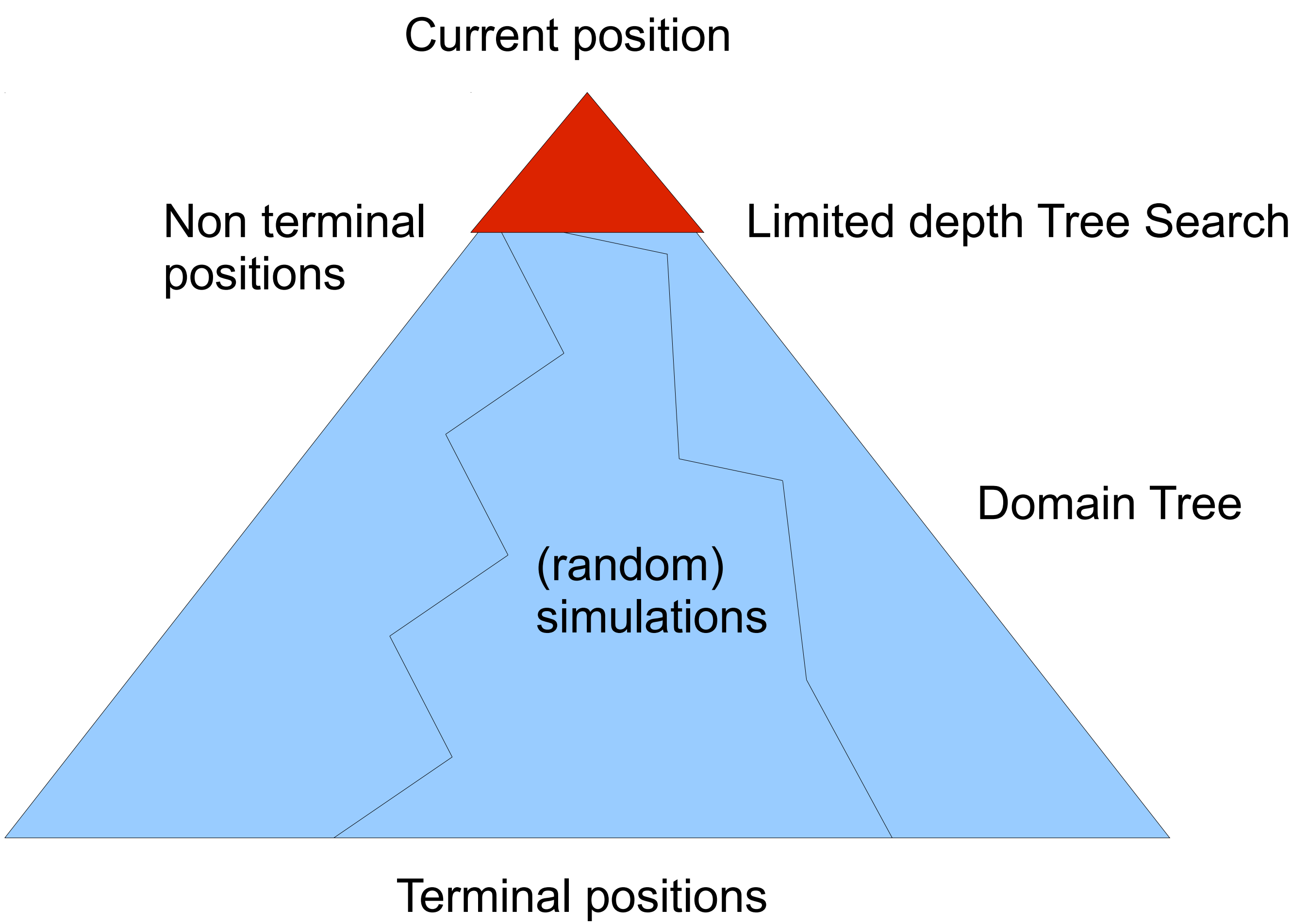
Overall approach

•



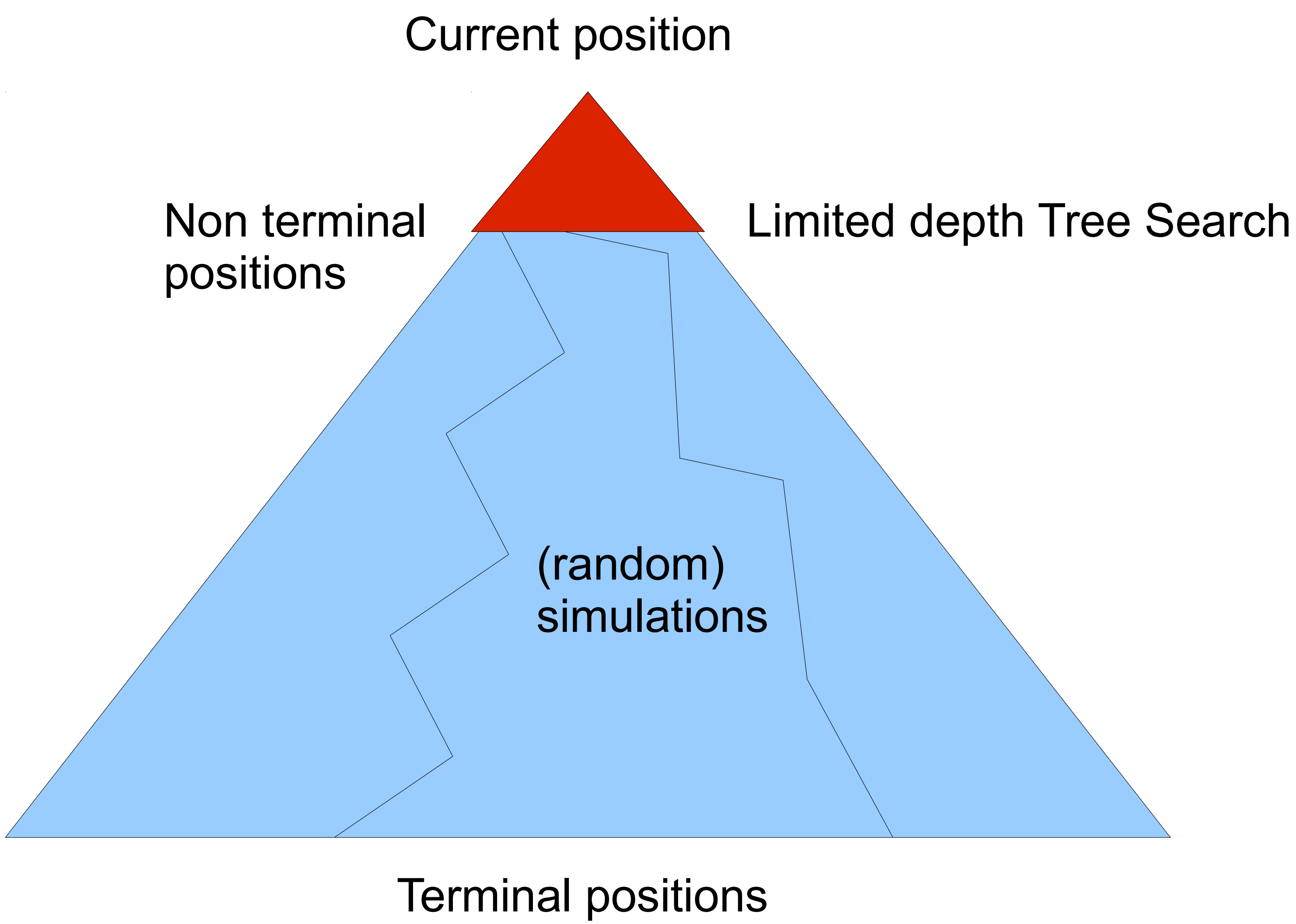
Why Monte-Carlo ?

- Evaluating a non terminal position :
  - a lot of evaluations of terminal positions reached with simulations
  - a knowledge-based evaluation function



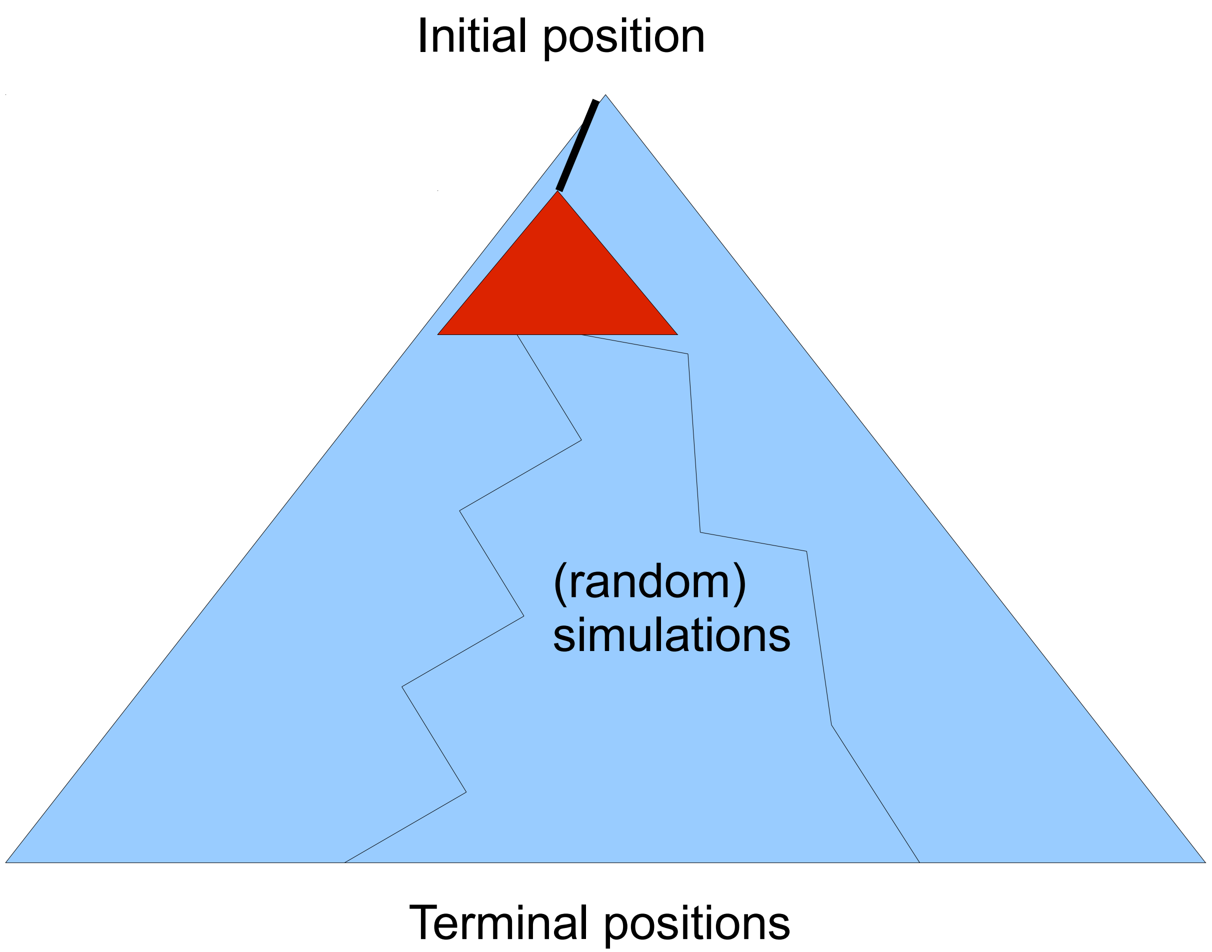
Why Monte-Carlo Tree Search ?

- Choosing the next action to perform must be accurate.
- 2-player games, n-player games, 1-player game (planning).



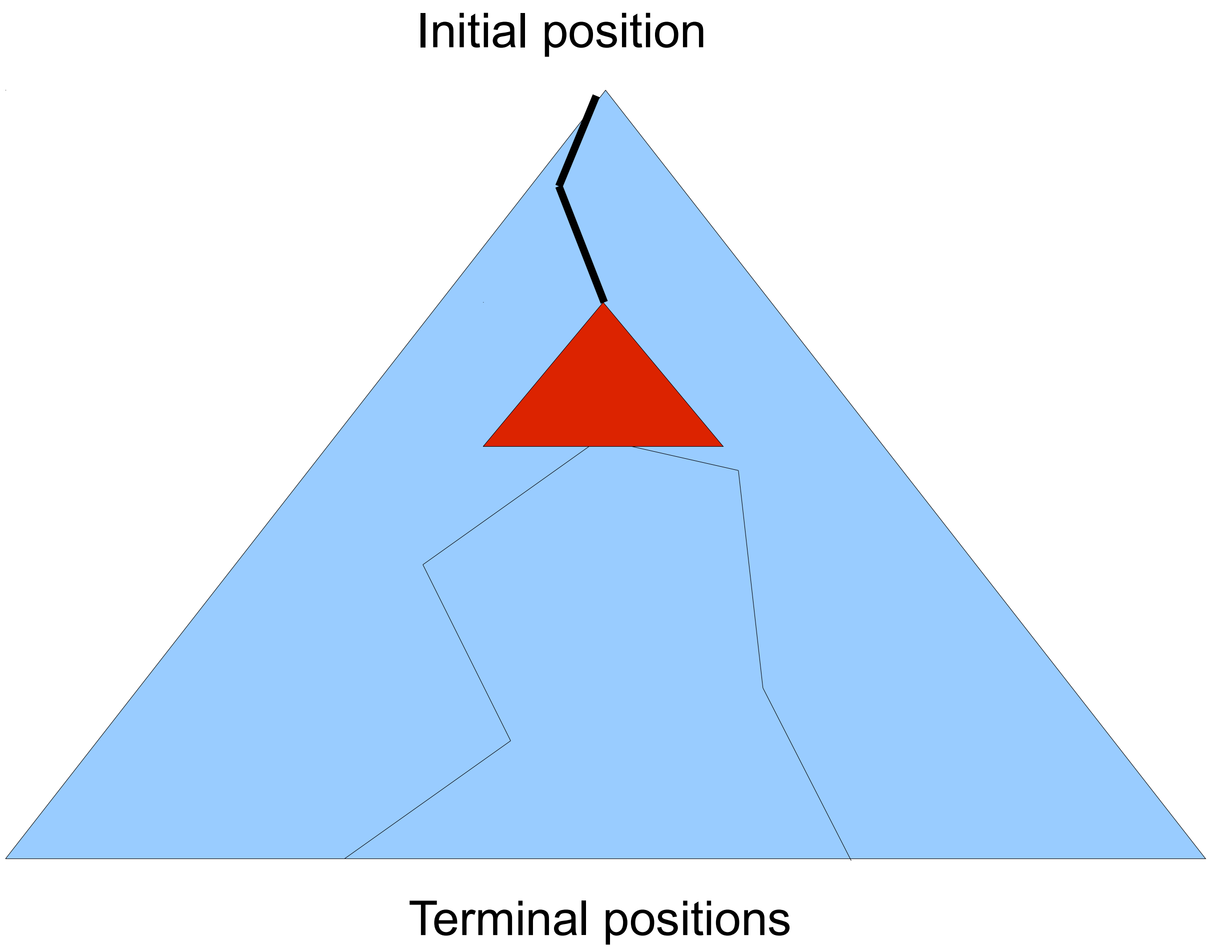
Sequential decision making

- The first action is performed...
- … another MCTS is launched to choose the next action...



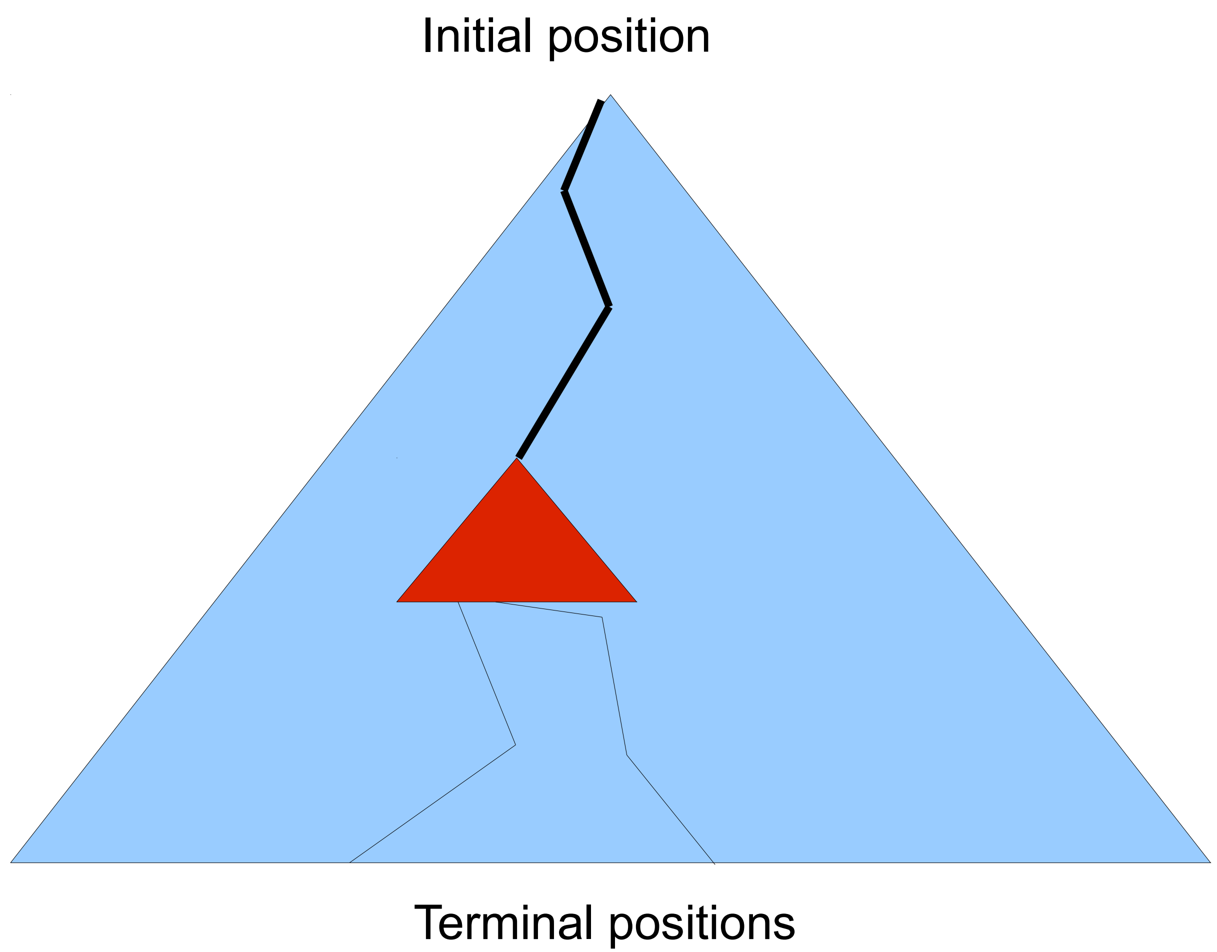
Sequential decision making

- Then the second one is performed...



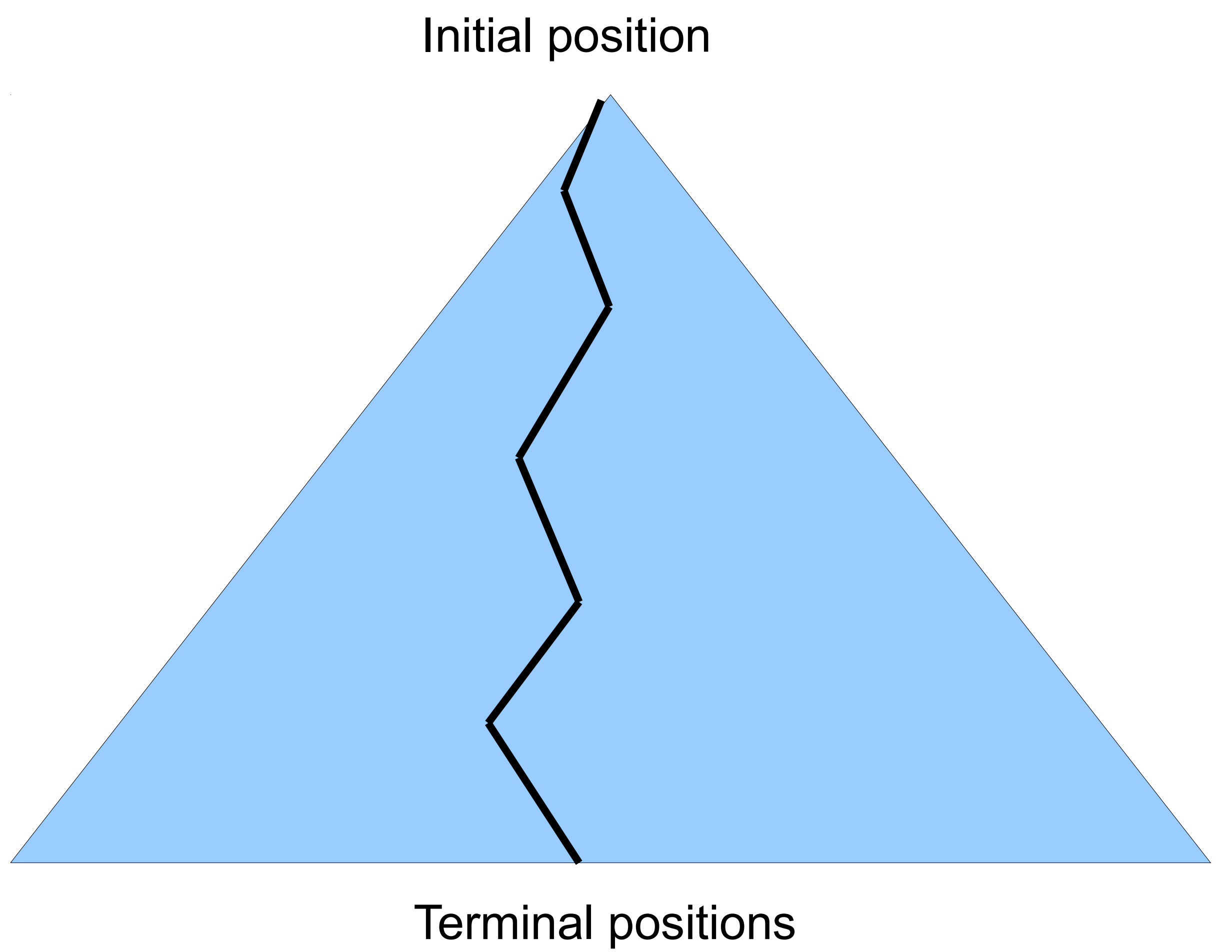
Sequential decision making

- And so on...



Sequential decision making

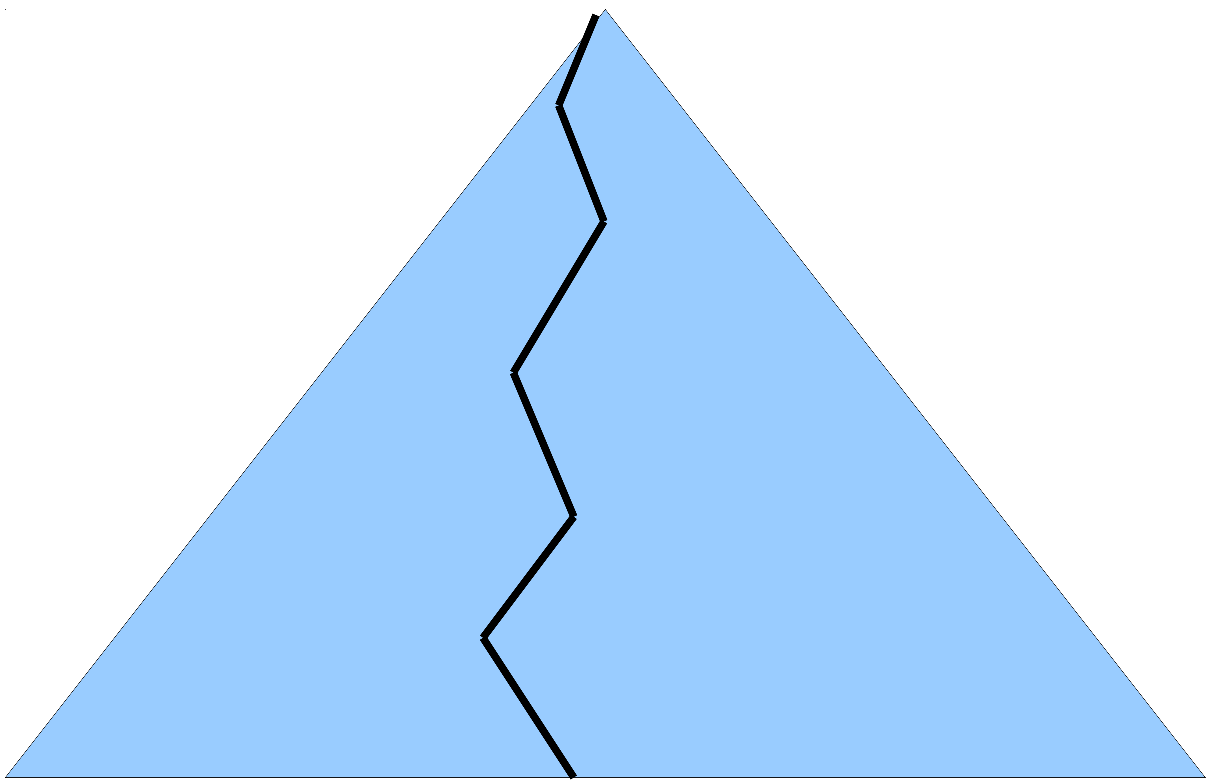
- Until the sequence reaches the end of the domain.



Level one simulation

- Level 1 simulation – or « smart » simulation -
- … usable within another MCTS

Initial position



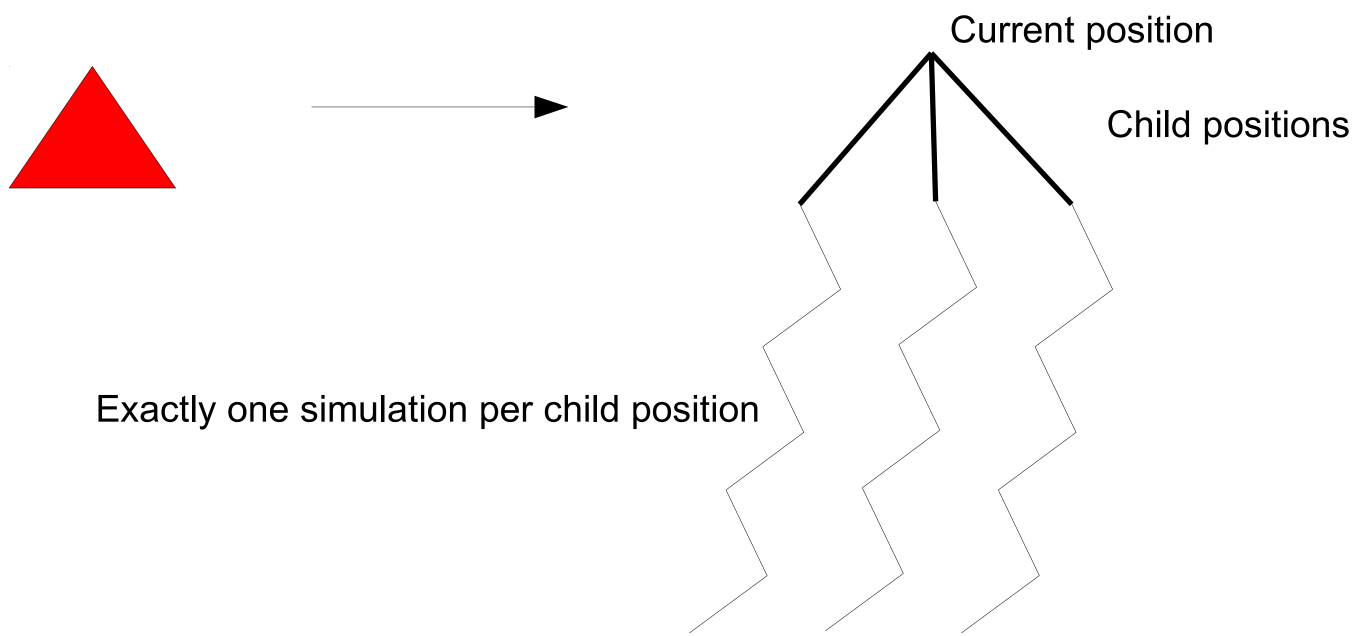
Terminal positions

Nested MCTS

- Level N MCTS performs a simulation by using level N-1 simulations
- Level 0 simulations :
  - Random simulations
  - Pseudo-random simulations
- MCTS is too costly.
- Is it possible to design an algorithm that could be nested several times ?

(Nested) MCS (T has vanished...)

- Depth-one greedy search using one simulation per child node (Cazenave 2009)



- The CPU time is less important
- The levels can be nested several times (2 or 3 or 4 depending on the domain)
- Planning, 1-player games, expression discovery, weak schur numbers, morpion solitaire, etc.

CPF Complexity

- Number of joint actions is exponential in the number of agents
  - 10 agents and 5 actions yield $5^{10} \approx 10^{7}$ joint actions

- « Best first » searches (MCTS, MCS, A*) do not work on the simple example

  … because of the branching factor

How to adapt the MCS approach ?

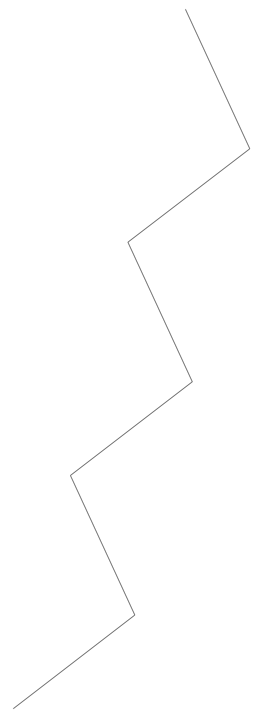
- Question 1 : How to choose a move at random and perform only one simulation ?

? ? ?

- Question 2 : How to develop a search without being stuck near the initial node ?

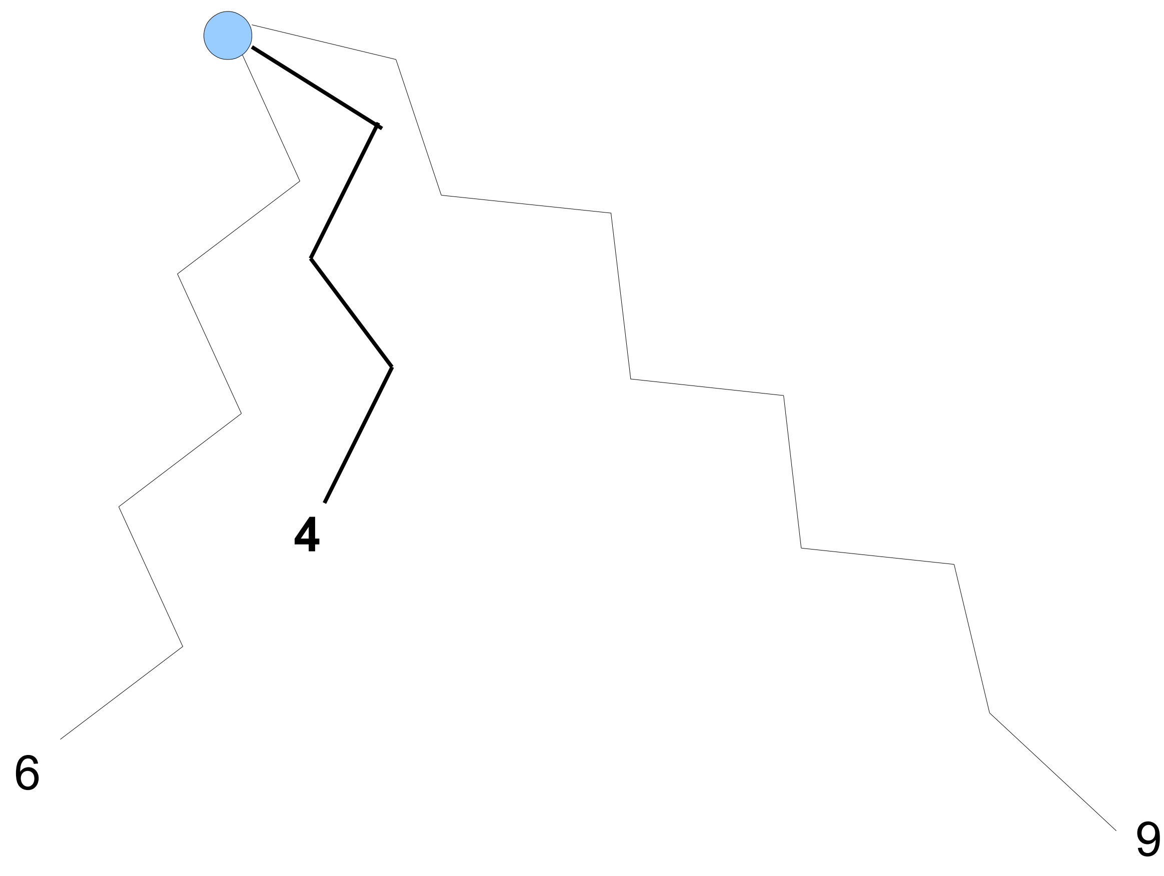
? ? ?

MC Fork Search rationale

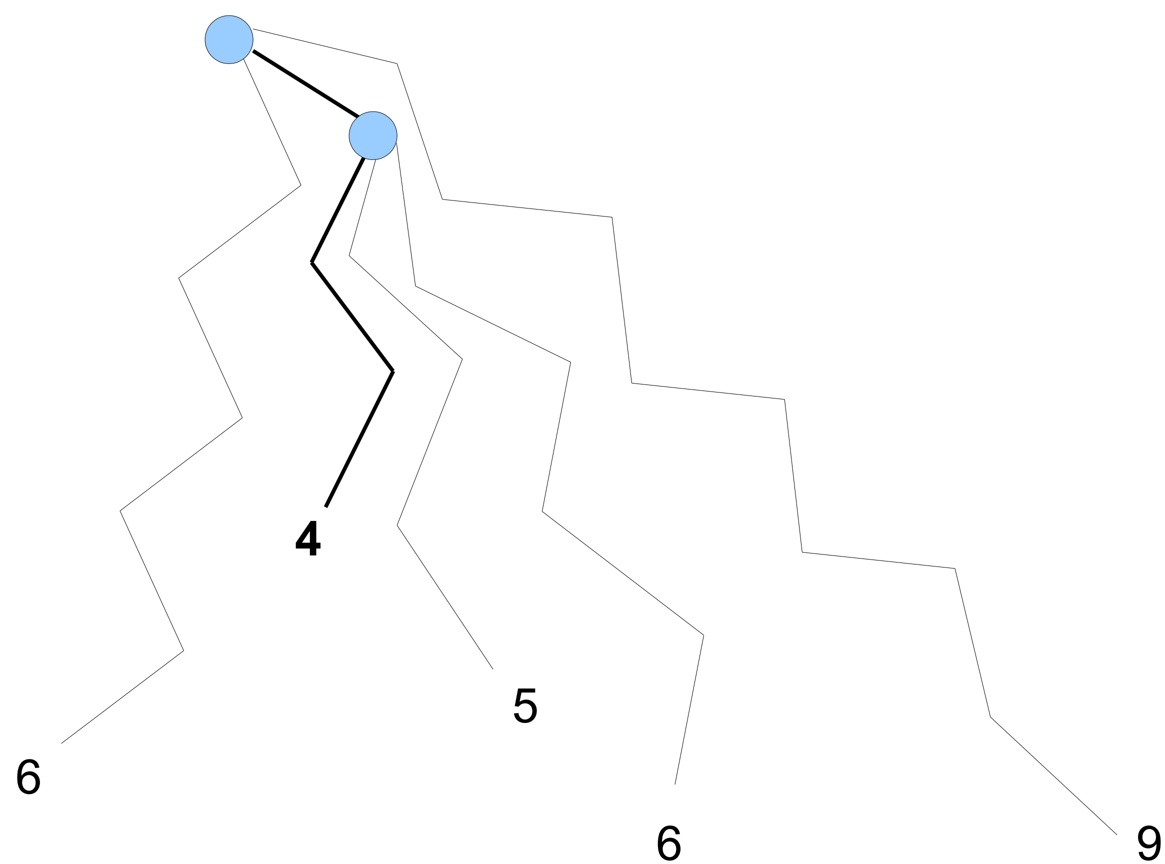


6

MC Fork Search rationale

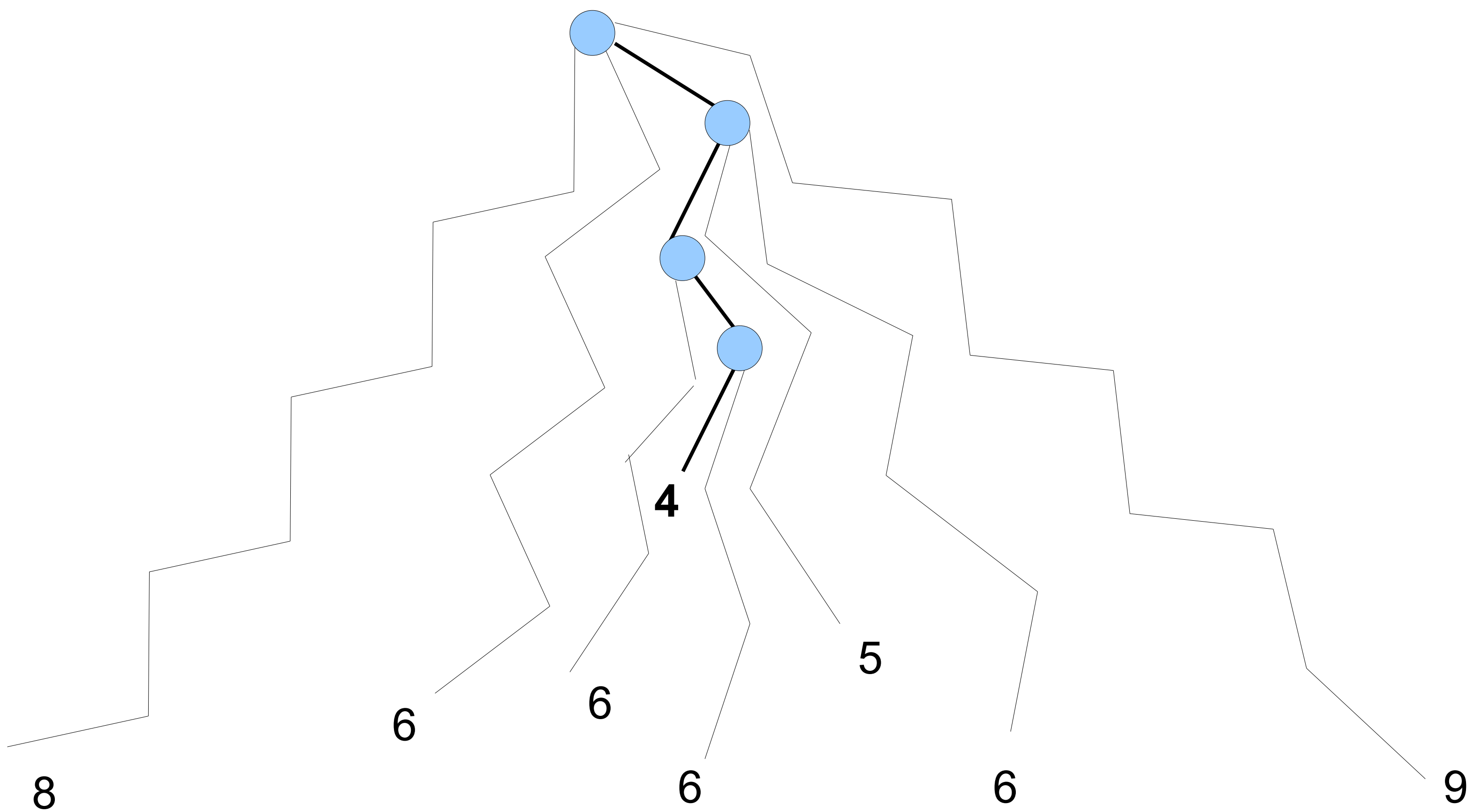


MC Fork Search rationale



MC Fork Search rationale

- fork the next simulation along the current best sequence



- Use UCB (or any other rule) on the whole tree
- No top down order

NMCFS pseudo-code

```
int NMCFS(start, goal, bestSeq, lev) {
    if (lev==0) then return sample(start, goal, bestSeq)
    n=1; lmin=+∞; actualSeq; Node root(a)
    while (n≤it) do {
        Node nd = root.selectNode(); pos=nd.position
        l = NMCFS(pos, b, actualSeq, lev-1)
        If (l+nd.depth<lmin) then {
            lmin = l+nd.depth
            bestSeq = seq(root(a), nd) + actualSeq
        }
        nd.backUp(l); nd.append(actualSeq,l,b); n=n+1
    }
    return lmin
}
```

UCB rule adapted to MCFS

$$\text{argmin}_{\text{builtTree}} \left(\text{lmin+depth} - C \left(\text{variance}\ \log(n)/(1+\text{nforks})\right)^{1/2}\right)$$

Selection is global

Exploitation focused on nodes belonging to the shortest sequences

Exploration focused on nodes with high variance and low number of forks

CPF simulations (1/2)

- Optimal individual paths are pre-computed (Botea 2011)

- Each agent knows its optimal elementary actions.
  - (1) Greedy choice
  - (2) Pseudo-random choice
- Each agent expresses a « wish » (the next cell)

- Collision management
  - Choose the agent that gives the way at random

- Iterative process to choose one joint action

CPF simulations (2/2)

- Choice of the joint action :

  While the joint action is not correct {

  Ask each agent for its wish

  Manage the collisions

  }

- Progressive relaxation:
  - (1) Optimal actions only
  - (2) Optimal actions + stay
  - (3) All actions (random walk)

Experiments

- Set of problems
  - (Khorshid 2011) Tnea
  - (Luna & Bekris 2011) Tnea
  - CPF N-puzzles (Yu 2012) Time
  - Many-agents-large-grid problems Tnea
- 3.2 Ghz CPU with 6 Gbytes memory
- Comparing results ?
  - Our work: optimizing 'Time'; (however 'Tnea' available)
  - Most of other work: optimizing 'Tnea'

Reference programs

- TASS (Khorshid 2011)
  - #agents = gridsize – 4
  - knowledge-based, fast, complete, sub-optimal.
- Push & Swap (Luna & Bekris 2011)
  - #agents = gridsize – 2
  - knowledge-based, fast, complete, sub-optimal.
- TOMPP (Yu & la valle 2012)
  - Network flow analogy, uses linear programming
  - Time-optimal.

Results on Khorshid's problems

Comparison with Tree-based Agent Swap Strategy (TASS)

	T		Tnea					
	MCFS	Optim	**TASS**	**MCFS**	Optim	time	lev	it
520	6	6	34	**17**	17	0.1s	2	5
519	8	8	30	**18**	12	0.03s	2	5
518	10	10	58	**26**	≤26	2m	2	30
517	13	13	170	**31**	≤31	3m	2	30
515	15	≤15	459	**71**	≤71	30m	3	40
516	**19**	≤18	234	**86**	**≤69**	72h	3	30

Results on Luna & Bekris problems

Comparison with Push & Swap

	T		nea					
	MCFS	Optim	**Push&swap**	**MCFS**	Optim	time	lev	it
rotation	1	1	18	**16**	16	0.01s	**1**	**1**
tree	6	6	18	**12**	12	0.01s	1	5
string	8	8	26	**20**	≤20	0.02s	1	5
corner	8	8	50	**32**	≤32	1s	2	5
connection	16	≤16	86	**70**	≤70	1m	2	20
tunnel	15	≤15	81	**49**	≤49	1h	3	30
loopchain	19	≤17	350	**106**	≤98	12h	2	200

Results on N-puzzles with no hole

- N=8, 15, 24: one hole; N=9, 16, 25: no hole.

N	Branch. factor	T	nea	time	level	it
8	123	4	26	0.1s	1	10
9	27	6	38	5s	1	10
15	3815	7	84	20m	3	50
16	951	8	90	4h	3	50
24	$\approx 10^5$	7	141	8h	3	30
25	$\approx 3\ 10^4$	8	120	30h	3	30

Result on a 25-puzzle (from Yu 2012)

- T=0
- Tnea=0

Position

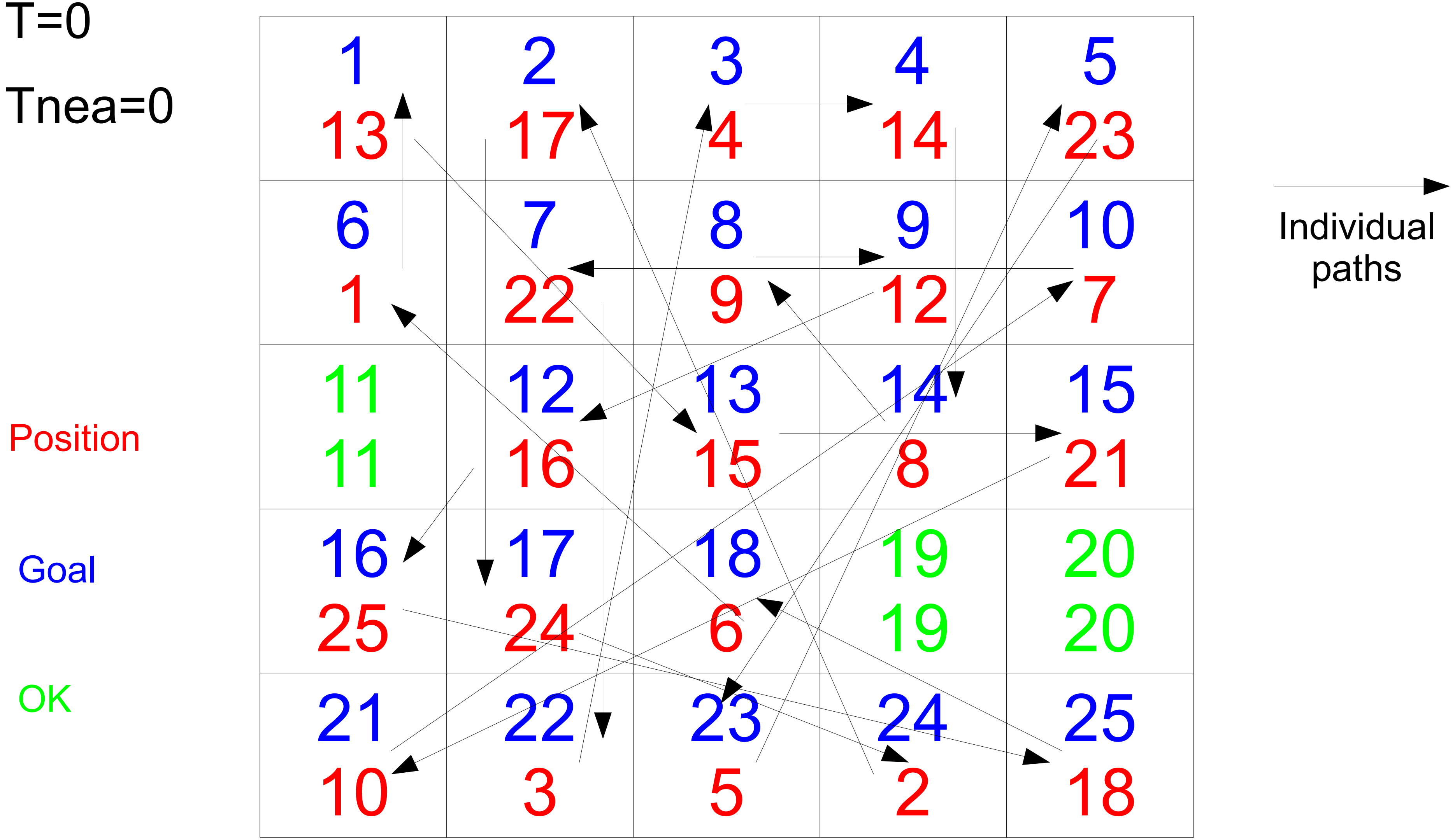
Goal

OK

1 13	2 17	3 4	4 14	5 23
6 1	7 22	8 9	9 12	10 7
11 11	12 16	13 15	14 8	15 21
16 25	17 24	18 6	19 19	20 20
21 10	22 3	23 5	24 2	25 18

Result on a 25-puzzle (from Yu 2012)

- T=0
- Tnea=0



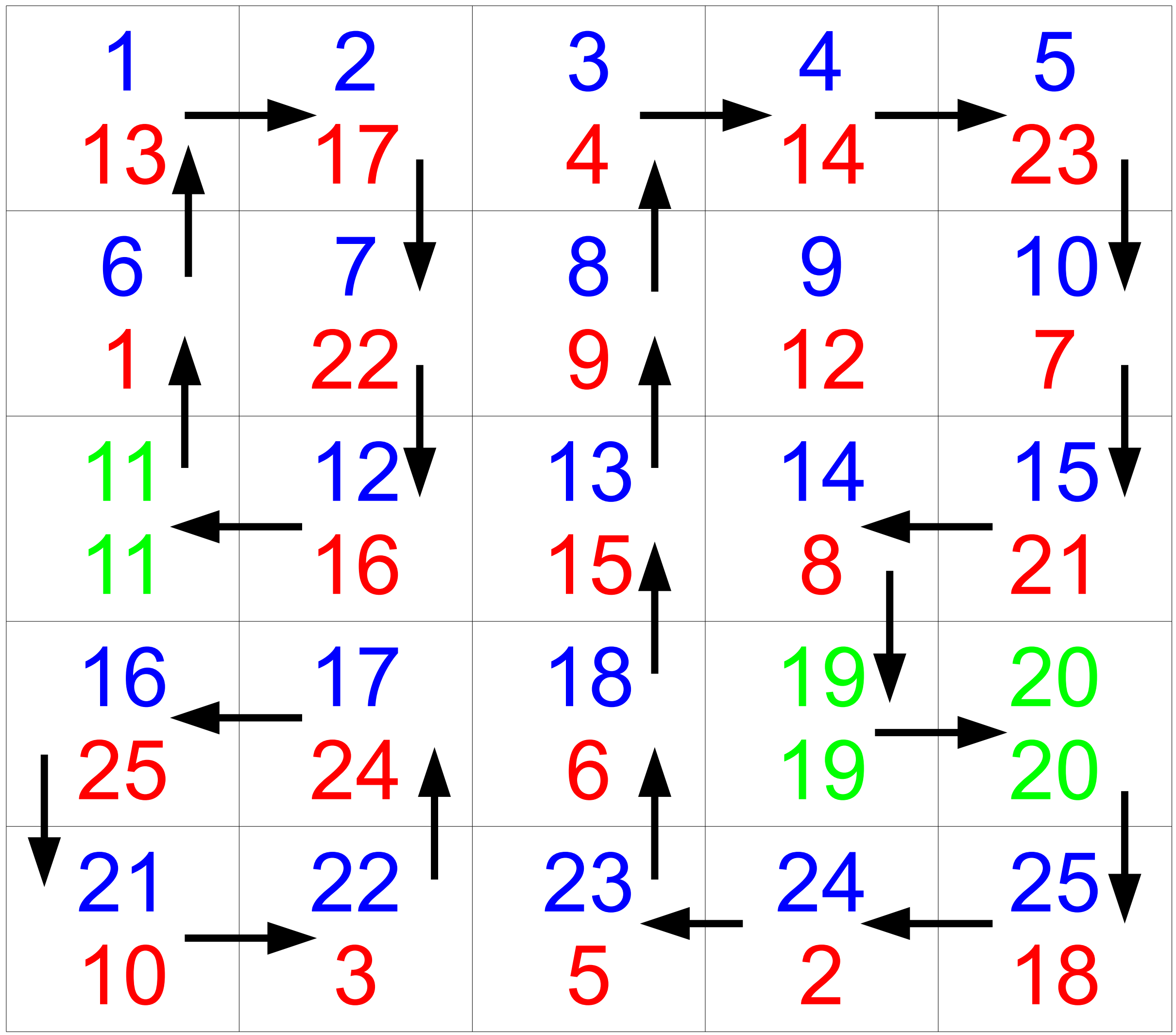
A joint action = several cycles

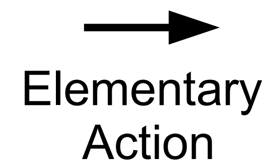
- T=0
- Tnea=0
- nea=24

Position

Goal

OK





Result on a 25-puzzle

- T=1
- Tnea=24
- nea=10

Position

Goal

OK

1 1	2 13	3 9	4 4	5 14
6 11	7 17	8 15	9 12	10 23
11 16	12 22	13 6	14 21	15 7
16 24	17 3	18 5	19 8	20 19
21 25	22 10	23 2	24 18	25 20

Elementary Action

Result on a 25-puzzle

- T=2
- Tnea=34
- nea=10

Position

Goal

OK

1 1	2 13	3 9	4 4	5 14
6 11	7 17	8 6	9 15	10 12
11 16	12 22	13 5	14 21	15 23
16 24	17 3	18 2	19 8	20 7
21 25	22 10	23 18	24 20	25 19

Elementary Action

Result on a 25-puzzle

- T=3
- Tnea=44
- nea=14

1 1	2 13	3 9	4 4	5 14
6 11	7 17	8 6	9 15	10 12
11 22	12 3	13 2	14 5	15 23
16 16	17 10	18 8	19 21	20 7
21 24	22 25	23 18	24 20	25 19

Position

Goal

OK

Elementary Action

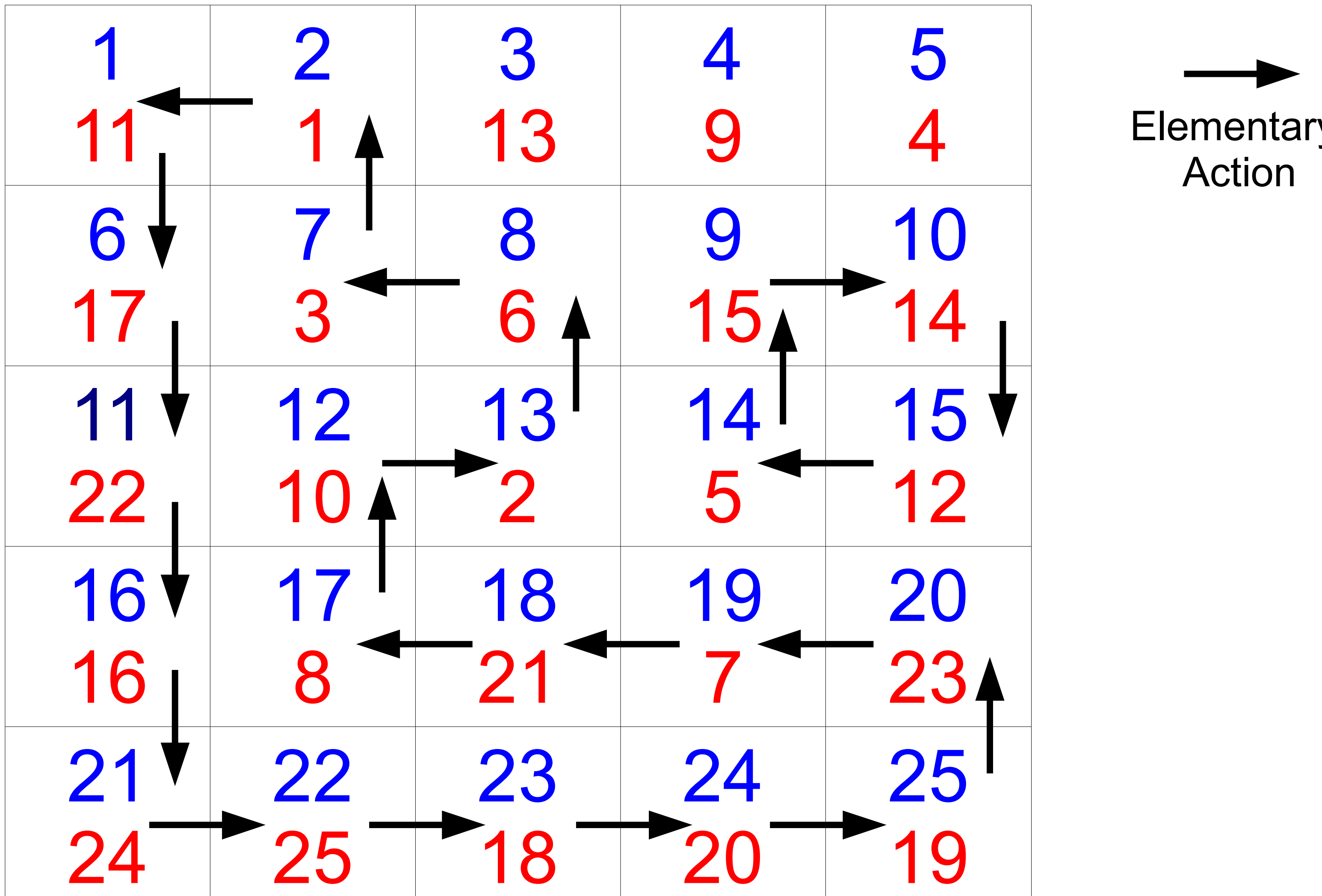
Result on a 25-puzzle

- T=4
- Tnea=58
- nea=22

Position

Goal

OK



Result on a 25-puzzle

- T=5
- Tnea=80
- nea=12

1 1	2 3	3 13	4 9	5 4
6 11	7 6	8 2	9 5	10 15
11 17	12 8	13 10	14 12	15 14
16 22	17 21	18 7	19 23	20 19
21 16	22 24	23 25	24 18	25 20

Elementary Action

Position

Goal

OK

Result on a 25-puzzle

- T=6
- Tnea=92
- nea=20

1 1	2 3	3 13	4 9	5 4
6 6	7 2	8 10	9 5	10 15
11 11	12 8	13 7	14 12	15 14
16 17	17 22	18 23	19 18	20 19
21 16	22 21	23 24	24 25	25 20

Elementary Action

Position

Goal

OK

Result on a 25-puzzle

- T=7
- Tnea=112
- nea=8

1 1	2 2	3 3	4 9	5 4
6 6	7 8	8 13	9 10	10 5
11 11	12 7	13 12	14 14	15 15
16 16	17 17	18 18	19 19	20 20
21 21	22 22	23 23	24 24	25 25

Position

Goal

OK

Elementary Action

Result on a 25-puzzle

- T=8
- Tnea=120
- nea=0

1 1	2 2	3 3	4 4	5 5
6 6	7 7	8 8	9 9	10 10
11 11	12 12	13 13	14 14	15 15
16 16	17 17	18 18	19 19	20 20
21 21	22 22	23 23	24 24	25 25

$\longrightarrow$

Elementary Action

Position

Goal

OK

Many-agents-large-grid result

-

gridsize	#agents	#obst.	pb	T	h	nea	time	lev	it
8 x 8	10	15	10	**12**	12	63	0.1s	1	5
			11	**11**	11	48			
25 x 25	100	125	1	43	41	2351	6h	2	50
			2	**43**	43	2508			
			3	44	41	2460			
			4	44	40	2569			
100 x 100	100	1000	1	**164**	164	6879	1m	1	1
	200	2000	1	**165**	165	14203	5m	1	1
	400	2000	1	**171**	171	33286	15m	1	1
200 x 200	400	4000	1	**344**	344	55620	1h	1	1

Many-agents-large-grid limits

- Main limit:
  - computation and storage of distances between cells

- For 200x200 grids:
  - Memory size in $200^4 = 1.6\ 10^9$
  - Pre-computation of distances between cells
  - Video game benchmarks: 500x500 grids (Sturtevant 2012)

- Other limit: Time for running one simulation

Algorithm features

- Memory use
  - No problem with the nested version
- Completeness
  - Worst case: the meta-graph is browsed by a meta-random walk
  - Solvable problem = meta-graph connected = random walk reaches all states
- Anytime
- Near-optimality
  - $h = \max_{agent} \mathrm{pathLength}(agent)$
  - No other measure of distance to optimality

Conclusions and perspectives

- Nested MCFS solves CPF problems
  - Not solved by classical solvers (A*, MCTS, MCS)
  - N-puzzles without hole, Khorshid, Luna&Bekris, Many-agents
- MCFS is not sensitive to the branching factor
- Features: anytime, complete, near-optimal
- Future work
  - How to set #level and #fork automatically ?
  - Test on other very-high branching factor problems
  - CPF: 8-connectivity, larger grids
  - Diameter of the graph of the nxn-puzzle
  - 2-player games ?

Thank you for your attention!

- Questions ?
- bruno.bouzy@parisdescartes.fr