Simple Direct Propositional Encoding of Cooperative Path Finding Simplified Yet More

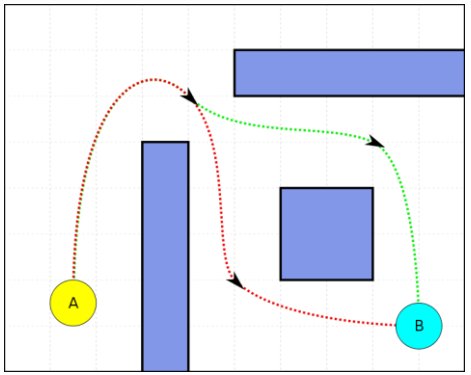
Pavel Surynek

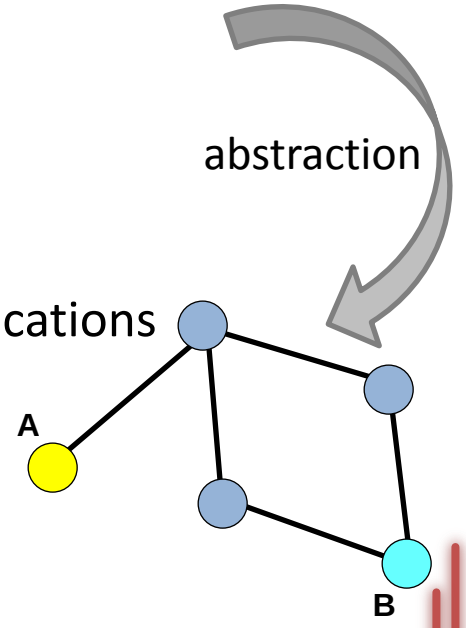
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic

MICAI 2014, Tuxtla Gutiérrez, Mexico

Cooperative Path-Finding (CPF)

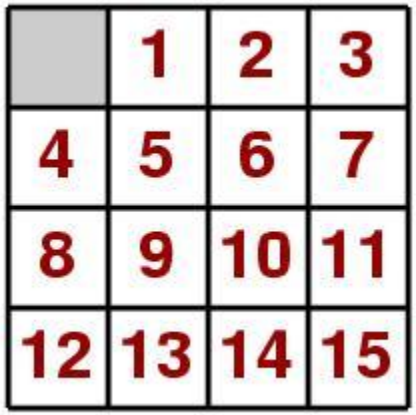
- Agents can **move only**
  - each agent needs to relocate itself
  - initial and goal location
- **Physical limitations**
  - agents must **not collide** with each other
  - must avoid **obstacles**
- **Abstraction**
  - environment – **undirected graph** G=(V,E)
    - vertices V – **locations** in the environment
    - edges E – **passable** region between neighboring locations
  - agents – items placed in vertices
    - **at most one** agents per vertex
    - **at least one** vertex empty to allow movements

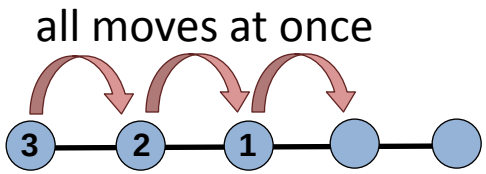




CPF Formally

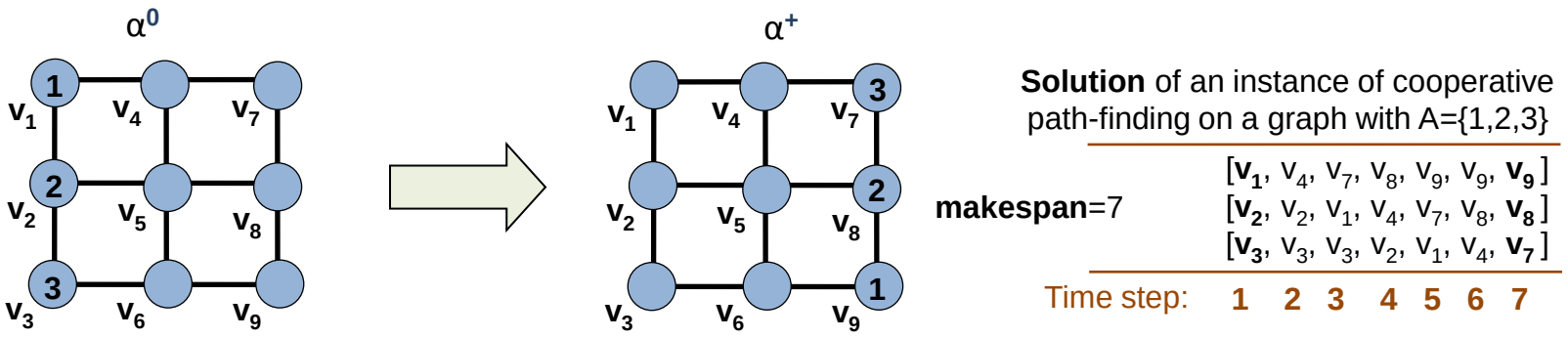
- A **quadruple** $(G, A, \alpha^0, \alpha^+)$, where
  - $G=(V,E)$ is an **undirected graph**
  - $A = \{a_1,a_2,...,a_\mu\}$, where $\mu<|V|$ is a **set of agents**
  - α^0: $A \rightarrow V$ is an **initial arrangement of agents**
    - uniquely invertible function
  - α^+: $A \rightarrow V$ is a **goal arrangement of agents**
    - uniquely invertible function
- **Time** is discrete – time steps
- **Moves/dynamicity**
  - depends on the model
  - **agent moves** into unoccupied neighbor
    - no other agent is entering the same target
  - sometimes **train-like** movement is allowed
    - only the leader needs to enter unoccupied vertex





Solution to CPF

- **Solution** of (G, A, α^0, α^+)
  - sequence of arrangements of agents
  - *(i+1)*-th arrangement obtained from *i*-th by legal moves
  - **the first arrangement** determined by α^0
  - **the last arrangement** determined by α^+
    - all the agents in their goal locations
- The length of solution sequence **= makespan**
  - **optimal/sub-optimal** makespan



Solution of an instance of cooperative path-finding on a graph with A={1,2,3}

makespan=7

$[\mathbf{v_1}, v_4, v_7, v_8, v_9, v_9, \mathbf{v_9}]$
$[\mathbf{v_2}, v_2, v_1, v_4, v_7, v_8, \mathbf{v_8}]$
$[\mathbf{v_3}, v_3, v_3, v_2, v_1, v_4, \mathbf{v_7}]$

Time step: **1 2 3 4 5 6 7**

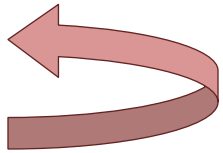
Motivation for CPF

- **Container rearrangement**
  (agent = container)
- **Heavy traffic**
  (agent = automobile (in jam))
- **Data transfer**
  (agent = data packet)
- **Ship avoidance**
  (agent = ship)

CPF as SAT

- **SAT = propositional satisfiability**
  - a formula ϕ over 0/1 (false/true) variables
  - Is there a valuation under which ϕ evaluates to 1/true?
    - **NP-complete** problem
- **SAT solving** and **CPF**
  - powerful SAT solvers
    - MiniSAT, clasp, glucose, glue-MiniSAT, crypto-MiniSAT, …
    - intelligent search, learning, restarts, heuristics, …
  - **CPF $\Rightarrow$ SAT**
    - all the advanced techniques **accessed almost for free**
- **Translation**
  - given a CPF $\Sigma=(G, A, \alpha^0, \alpha^+)$ and a **makespan η**
  - **construct** a formula ϕ
    - satisfiable iff Σ has a solution of makespan η

$(x \vee \neg y) \wedge (\neg x \vee y)$
Satisfied for x = 1, y = 1

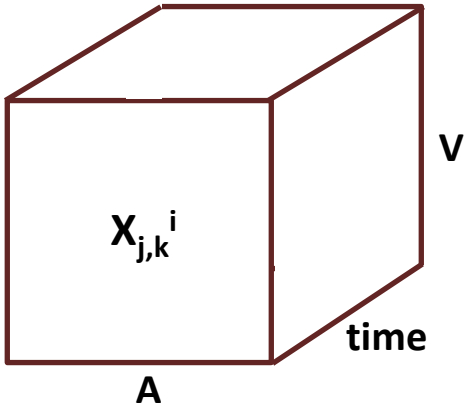


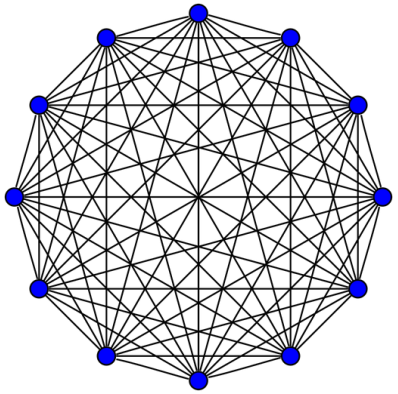
Encoding of CPF

- **How to encode** a question if there is a solution of makespan ***η*?**
  - Build **time expansion network**
    - Represent arrangements of agents at steps 1,2…,η
      - **step 1** … α^0
      - **step η** … α^+
    - Encode **dynamicity** of CPF (valid transitions)
      - consecutive arrangements must be obtainable by valid moves
- Encoding design issues
  - **(i)** suggest propositional variables
    - represent arrangement of agents in graph G over time steps 1,2…,η
  - **(ii)** introduce constraints (clauses)
    - remove **non-arrangements** (more than one agent in a vertex)
    - remove **invalid transitions** (agents collide)

DIRECT Encoding of CPF

- The design of **propositional variables**
  - recall what we need to model
    - **A** = $\{a_1, a_2, \ldots, a_\mu\}$
      - a set of agents
    - **V**=$\{v_1, v_2, \ldots, v_n\}$
      - a set of vertices
    - **time steps** 1,2…,η
  - $\mathbf{X_{j,k}^i} \in \{true, false\}$
    - *TRUE* iff agent a_k appears in v_j at time step i
      - allow to represent invalid states

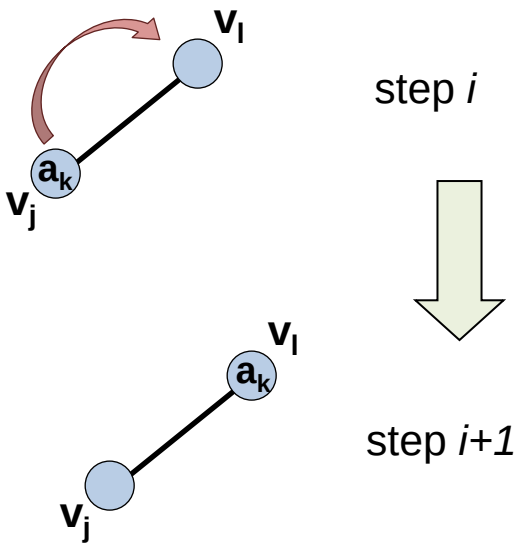




- The design of **constraints**
  - rule out invalid states (non-arrangements)
  - enforce valid transitions between time steps
    - many binary clauses
      - **at most one** agent is placed in a vertex at each time step
      - support **unit propagation**

Auxiliary Variables

- Auxiliary variables allow to build the CNF formula in a **hierarchical manner**
  - **relocation** of agent a_k
    - at time step i
    - from vertex v_j to vertex v_l
  - **target** v_l
    - must be empty at time step i
  - **source** v_j
    - must be empty at time step $i+1$



- relocation in terms of clauses
  - $X_{j,k}^{\,i} \wedge X_{l,k}^{\,i} \Rightarrow (\bigwedge_{h=1}^{\mu} \neg X_{j,h}^{\,i}) \wedge (\bigwedge_{h=1}^{\mu} \neg X_{l,h}^{\,i+1})$

relocation occurred — target v_j is empty at i — source v_l is empty at $i+1$

  - emptiness constraints are the same for all the agents

SIMPLIFIED Encoding

- Repeating sub-formulae can be replaced with auxiliary variable
  - $X_{j,k}^{i} \wedge X_{l,k}^{i} \Rightarrow (\bigwedge_{h=1}^{\mu} \neg X_{j,h}^{i}) \wedge (\bigwedge_{h=1}^{\mu} \neg X_{l,h}^{i+1})$

    independent of relocated agent a_k — independent of relocated agent a_k
    - develops into **2μ ternary clauses**
- Introduce auxiliary propositional variable $\mathbf{E_j^i} \in \{true, false\}$
  - *TRUE* iff vertex v_j is empty at time step *i*
  - replace original constraint with $X_{j,k}^{i} \wedge X_{l,k}^{i} \Rightarrow E_j^i \wedge E_l^{i+1}$
    - develops into **2 ternary clauses**
- Introduce the **meaning** of auxiliary variables
  - $E_j^i \Rightarrow (\bigwedge_{h=1}^{\mu} \neg X_{j,h}^{i})$
    - develops into **μ binary clauses**
- **Fewer clauses but more decision variables**

Encoding Size Evaluation

- Comparison with **previous encodings**
  - **INVERSE** [Surynek, PRICAI 2012]
    - based on bit-vectors
    - comparison with domain independent encodings from SATPlan [Kautz, Selman, 1999] and SASE encoding [Huang, Chen, Zhang, 2010]
  - **ALL-DIFFERENT** [Surynek, ICTAI 2012]
    - based on bit-vectors and the all-different constraint
  - **DIRECT**
    - only the **decision variables** (no auxiliary ones)

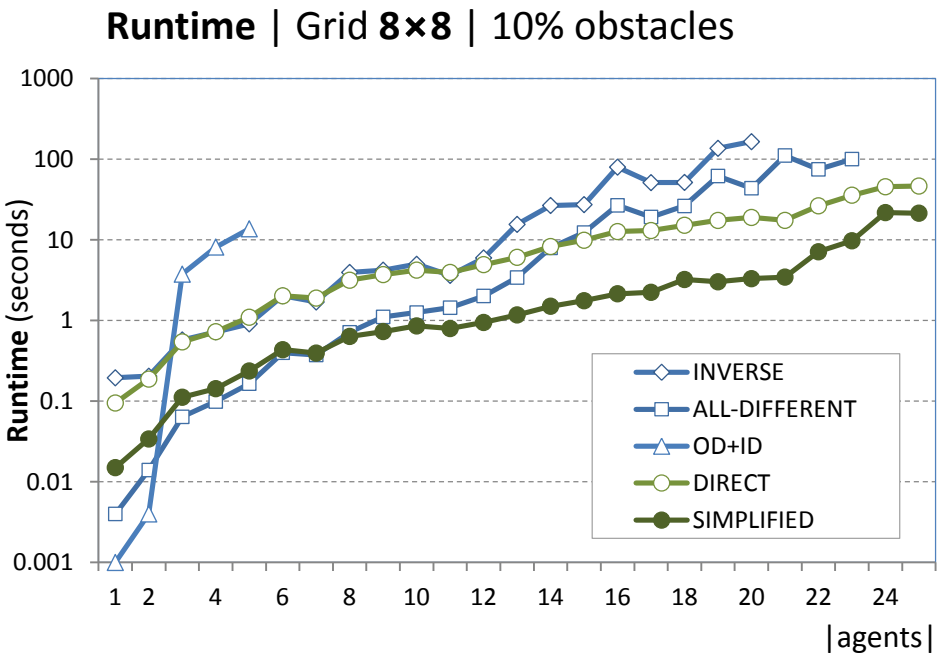
Setup: 4-connected grid, random initial arrangement and goal, 10% obstacles

16 time steps

Grid 8×8 |Agents|			INVERSE		ALL-DIFFERENT		DIRECT		**SIMPLIFIED**	
1	#Variables	Ratio	8 358.7	3.748	1 489.3	5.325	**814.4**	**28.539**	1 628.8	2.078
	#Clauses	Length	31 327.9	2.616	7 930.4	3.057	23 241.9	**2.149**	**3 384.6**	2.550
4			10 019.5	5.532	7 834.5	4.440	3 257.6	**35.589**	4 072.0	4.420
			55 437.0	2.641	34 781.9	3.103	115 934.3	**2.272**	**17 997.8**	2.374
16			11 680.3	7.820	67 088.3	3.231	13 030.4	**64.506**	13 844.8	10.853
			91 344.5	3.127	216 745.4	3.147	840 540.6	2.505	**150 259.2**	**2.180**
32			12 510.7	9.765	230 753.0	2.802	26 060.8	**105.084**	26 875.2	19.002
			122 170.3	3.733	646 616.2	3.168	2 738 584.7	2.621	**510 672.1**	**2.111**

Runtime Evaluation

- Comparison with **previous encodings** + A*-based **ID+OD** [Standley, IJCAI 2011]
  - same setup as in the size evaluation



Grid 8×8 |Agents|	1	2	4	8	12	16	20	24
Makespan	6.4	6.1	8.1	10.5	9.8	11.0	11.9	12.7

- SIMPLIFIED encoding performs as best for **higher number** agents

Conclusions and Observations

- **CPF as SAT**
  - **Advantages**
    - search techniques
      - advanced search techniques from SAT solvers accessed
    - modularity
      - exchangeable modules – SAT solver, encoding
  - **Disadvantages**
    - energy extensive solutions
      - agents move too much
- **SIMPLIFIED** Encoding
  - **space efficient**
    - small number of variables and clauses
  - **time efficient**
    - can be solved faster than previous encodings
    - SAT-based approach with SIMPLIFIED encoding outperforms A*-based approach