Reduced Time-Expansion Graphs and Goal Decomposition for Solving Cooperative Path Finding Sub-Optimally

Pavel Surynek
Charles University in Prague, Faculty of Mathematics and Physics, Czech Republic
pavel.surynek@mff.cuni.cz

Abstract

Solving cooperative path finding (CPF) by translating it to propositional satisfiability represents a viable option in highly constrained situations. The task in CPF is to relocate agents from their initial positions to given goals in a collision free manner. In this paper, we propose a reduced time expansion that is focused on makespan sub-optimal solving. The suggested reduced time expansion is especially beneficial in conjunction with a goal decomposition where agents are relocated one by one.

1 Introduction and Motivation

The problem of cooperative path-finding (CPF) [Kornhauser *et al.*, 1984; Silver, 2005, Ryan, 2008] is a graph theoretical abstraction for many real life problems where the task is to cooperatively relocate a group of robots or other movable objects in a collision free manner. Each agent of the group is given its initial and goal position in the environment. The problem consists in constructing a spatial temporal plan for each agent by which it can relocate from its initial position to the given goal. The environment where agents move is modeled as an undirected graph [Kornhauser *et al.*, 1984] where vertices represent locations and edges represent possibility of relocation between two locations.

Agents are represented as abstract items placed in vertices while at most one agent is located in each vertex. An agent can instantaneously relocate itself to the neighboring vertex assumed the target vertex is unoccupied and no other agent is trying to enter the same target vertex.

In this research, we further develop solving of CPF by translating it to propositional satisfiability (SAT) [Biere *et al.*, 2009]. Recent propositional encodings [Surynek, 2012a, 2012b, 2013, 2014] of CPF are based on time expansion of the graph modeling the environment so that the encoding is able to represent arrangements of agents over the graph at all the time steps up to the final one. Since there may be many time-steps before all the agents reach their goals, these encodings may become extremely large and hence unsolvable in reasonable time. We are trying to overcome this limitation by reducing the expansion of the graph in this work.

1.1 Context of Related Works

The approach to solve CPF by reducing it to SAT has multiple alternatives. There exist algorithms based on search that find makespan optimal or near optimal solutions. The seminal work in this category is represented by Silver's WHCA* algorithm [Silver, 2005]. Recent contributions include OD+ID [Standley and Korf, 2011], which is a combination of A* and powerful agent independence detection heuristics, and ICTS [Sharon et al., 2013] which employs the concept of increasing cost tree (instead of makespan, the total cost of solution is optimized). Other approaches resolve conflicts among robot trajectories when avoidance is necessary [Čáp *et al.*, 2013; Barer et al., 2014; Wagner and Choset, 2015].

Fast polynomial time algorithms for generating makespan suboptimal solutions include PUSH-AND-ROTATE [de Wilde *et al.*, 2014]. The drawback of these algorithms is that their solutions are dramatically far from the optimum.

Translation of CPF to a different formalism, namely to answer set programming (ASP), has been suggested in [Erdem *et al.*, 2013]. Integer programming (IP) as the target formalism has been also used [Yu and LaValle, 2013]. The choice of SAT as the target formalism is very common in domain independent planning where the idea of time expansion [Kautz and Selman, 1999; Huang *et al.*, 2010] and its reductions [Wehrle and Rintanen, 2007] are studied.

2 Formal Definition of CPF

An arbitrary **undirected graph** $G = (V, E)$ can be used to model the environment where agents are moving. The placement of agents in the environment is modeled by assigning them vertices of the graph. Let $A = \{a_1, a_2, ..., a_\mu\}$ be a finite set of *agents*, then, an arrangement of agents in vertices of graph G is fully described by a *location* function $\alpha: A \rightarrow V$. At most **one agent** can be located in each vertex; that is α is uniquely invertible.

Definition 1 (COOPERATIVE PATH FINDING). An instance of *cooperative path-finding* problem is a quadruple $\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$ where location functions α_0 and α_+ define the initial and the goal arrangement of a set of agents A in G respectively. □

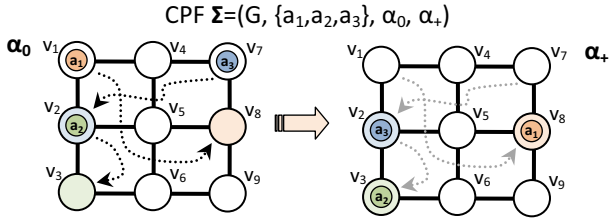
The dynamicity of the model supposes a discrete time divided into time steps. An arrangement α_i at the i-th time step can be transformed by a transition action which instantaneously moves agents in the non-colliding way to form a new arrangement α_{i+1}. The transition between α_i and α_{i+1} must satisfy the following *validity conditions*:

- $\forall a \in A$ either $\alpha_i(a) = \alpha_{i+1}(a)$ or $\{\alpha_i(a), \alpha_{i+1}(a)\} \in E$ (agents move along edges or not move at all), (1)
- $\forall a \in A \;\; \alpha_i(a) \neq \alpha_{i+1}(a) \Rightarrow \big(\forall b \in A \; \alpha_i(b) \neq \alpha_{i+1}(a)\big)$ (agents move to vacant vertices only), and (2)
- $\forall a, b \in A \;\; a \neq b \Rightarrow \alpha_{i+1}(a) \neq \alpha_{i+1}(b)$ (no two agents enter the same target/unique invertibility of resulting arrangement). (3)

The task in cooperative path finding is to transform α_0 using above valid transitions to α_+. An illustration of CPF and its solution is depicted in Figure 1.

Definition 2 (SOLUTION, MAKESPAN). A *solution* of a *makespan* m to a cooperative path finding instance $\Sigma = [G, A, \alpha_0, \alpha_+]$ is a sequence of arrangements $\vec{s} = [\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m]$ where $\alpha_m = \alpha_+$ and α_{i+1} is a result of valid transition from α_i for every $i = 1,2, \ldots, m-1$. □

It is known that finding makespan optimal solution to CPF is NP-hard [Ratner and Warmuth, 1986].



$\vec{s}$	α_0	α_1	α_2	α_3	$\alpha_4 = \alpha_+$
a_1	v_1	v_1	v_2	v_5	v_8
a_2	v_2	v_3	v_3	v_3	v_3
a_3	v_7	v_4	v_4	v_1	v_2

Figure 1. ***Cooperative path-finding (CPF)*** *on a 4-connected grid.* The task is to relocate three agents a_1, a_2, and a_3 to their goal vertices so that they do not collide with each other. A solution $\vec{s}$ of makespan 4 is shown.

3 (Sub)optimization in CPF via SAT

The approach we are suggesting here to obtain parameter optimal solutions is to employ *propositional satisfiability* (SAT) solving as the key technology. This approach has been already successfully applied in obtaining makespan optimal plans in domain-independent planning [Kautz and Selman, 1999; Huang *et al.*, 2010] as well as in CPF [Surynek, 2013].

In case of CPF, a propositional formula $F(\Sigma, \eta)$ such that it is satisfiable if and only if a given CPF Σ with makespan bound η is solvable can be constructed. Being able to construct such a formula $F(\Sigma, \eta)$ one can obtain the optimal makespan for the given CPF Σ by asking multiple queries whether formula $F(\Sigma, \eta)$ is satisfiable with different makespan bounds η.

Various strategies of the parameter for queries exist for getting the parameter optimal solution. The simplest is to try sequentially makespans $\eta = 1,2, \ldots$ until η is equal to the optimum (minimum). This strategy will be further referred as *sequential increasing*. Pseudo-code of the strategy is listed as Algorithm 1.

Algorithm 1. *SAT-based parameter optimal CPF solving – sequential increasing strategy.* The algorithm sequentially finds the smallest possible makespan η for that a propositional encoding of a given CPF $\Sigma = (G, A, \alpha_0, \alpha_+)$ is solvable.

input: Σ – a CPF instance
output: a pair consisting of the optimal parameter and corresponding parameter optimal solution

```
function Find-Optimal-Parameter (Σ = (G, A, α0, α+)): pair
1:  η ← 1
2   loop
3:      F(Σ, η) ← Encode-CPF-as-SAT (Σ, η)
4:      if Solve-SAT (F(Σ, η)) then
5:          let f be a satifying valuation of F(Σ, η)
6:          return (η, f)
7:      η ← η + 1
8:  return (∞, ∅)
```

4 Reduced Time Expansion Graph

The main drawback of makespan optimal CPF solving via SAT is the large size of the formulae that encode the optimization questions [Surynek, 2013, 2014]. The size of encoding formulae becomes especially prohibitive when they encode questions if a solution with a large makespan exists. This is due to the fact that existing encodings expands the graph modeling the environment over the time up to the given makespan bound η. At each time step of the expansion arrangement of agents over the graph is represented and constraints ensure that only transitions conforming to validity conditions are possible between arrangements at consecutive time steps.

Our idea hence was to **reduce** the time expansion with possible **relaxation** of the requirement of makespan optimality of the solution. The key observation is that if there is no need of any complex avoidance between agents (there is no need to visit a single vertex multiple times), no time expansion of the graph is necessary at all. The question if there is a solution (not necessarily makespan optimal) can be stated as a question of existence of vertex disjoint paths connecting initial positions of agents with their goals in the original graph. Translating of this question into SAT is possible as well.

Nevertheless, in real situations movement interactions among agents require complex avoidance. A single vertex may need to be visited multiple times. This led us to the suggestion of a concept of *reduced time expansion graph*, which combines the expansion reduction with ability to represent complex avoidance.

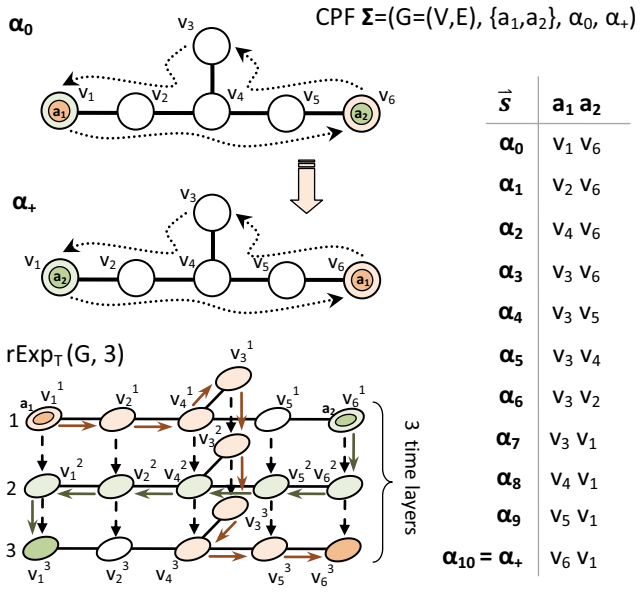


Figure 2. *An example of CPF and its solving through* ***reduced time expansion graph***. A reduced time expansion graph $\mathrm{rExp}_T(G,3)$ consisting of 3 time layers is build for a given CPF Σ. A solution to Σ corresponds to a collection of vertex disjoint paths connecting the initial positions agents in the first layer with their goal positions in the last time layer.

Definition 3 (REDUCED TIME EXPANSION GRAPH - $\mathrm{rExp}_T(G,\vartheta)$). Let $G=(V,E)$ be an undirected graph and $\vartheta \in \mathbb{N}$. A *reduced time expansion graph* with ϑ *time layers* associated with G is a directed graph $\mathrm{rExp}_T(G,\vartheta)=(V\times\{1,2,\dots,\vartheta\},E')$ where $E'=\{([u,l],\ [v,l])|\{u,v\}\in E; l=1,2,\dots,\vartheta\}\cup\{([v,l],[v,l+1])\mid l=1,2,\dots,\vartheta-1\}$. □

Note, that for each original undirected edge there are two directed arcs in both directions in the reduced time expansion graph. A *time-layer* in the reduced time expansion graph is an induced sub-graph of $\mathrm{rExp}_T(G,\vartheta)$ over the set of vertices $V\times\{l\}$ for a given $l\in\{1,2,\dots,\vartheta\}$.

Solving of CPF $\Sigma=[G,A,\alpha_0,\alpha_+]$ can be viewed as a search for *vertex disjoint paths* in $\mathrm{rExp}_T(G,\vartheta)$ that connect initial positions and goals in the first and the last time-layer respectively provided that the number of time-layers ϑ is sufficiently high. The idea is illustrated in Figure 2.

4.1 ϑ-RELAXED Propositional Encoding

The correspondence between the existence of vertex disjoint paths and the existence of a solution of CPF established in the previous section provides a guide how to design required propositional encoding. We merely need to design a propositional formula preferably in *conjunctive normal form* (CNF) [Biere *et al.*, 2009] that is satisfiable if and only if vertex disjoint paths connecting initial position and goals exist in $\mathrm{rExp}_T(G,\vartheta)$ for $\vartheta\in\mathbb{N}$.

Intuitivelly, the size and the structure of the resulting formula matters when it is solved by a SAT solver. Our choice was to design an encoding that is space efficient and contains short clauses. Note that short clauses support *unit propagation* [Biere *et al.*, 2009].

The encoding is separated into two parts. The first part is purely propositional and consists of variables that express selection of vertices and edges into paths – this can be also regarded as occupancy/selection of path by a flow of commodity. The inspiration for this design comes from the theory of network flows [Ahuja *et al.*, 1993]. The absence of necessity to distinguish between individual agents enables expressing the requirement that paths should be vertex disjoint as simple capacity constraints.

The distinguishable agents are treated in the second part of the model where a bit vector using binary encoding is associated with each vertex in $\mathrm{rExp}_T(G,\vartheta)$ to express what agent is occupying that. The benefit of using bit-vectors is that equality can be easily expressed over them. Both parts are put together by introducing a constraint that requires occupation by the same agent at both ends of a selected edges. Formally, the encoding – which we called ϑ-RELAXED – is introduced in the following definition.

Definition 4 (ϑ-RELAXED encoding - $F_{\vartheta-RE}(\Sigma)$). Let $\Sigma=[G,A,\alpha_0,\alpha_+]$ be a CPF with $G=(V,E)$. A *ϑ-RELAXED encoding* for CPF Σ consists of the following collections of variables for every time layer $l\in\{1,2,\dots,\vartheta\}$: finite domain variables $\mathcal{A}_v^l\in\{0,1,\dots,\mu\}$ for every $v\in V$ (that are encoded as bit vectors), propositional variables $\mathcal{X}_v^l$ for every $v\in V$, and propositional variables $\mathcal{E}_{u,v}^l$ for every ordered pair u,v such that $\{u,v\}\in E$ (that is, for a single edge $\{u,v\}\in E$ and l we have two propositional variables $\mathcal{E}_{u,v}^l$ and $\mathcal{E}_{v,u}^l$). Additionally, there is a set of propositional variables $\mathcal{E}_v^l$ for every every $v\in V$ and $l\in\{1,2,\dots,\vartheta-1\}$ representing interconnections between time layers. Constraints of ϑ-RELAXED encoding are as follows:

- $\mathcal{A}_v^l\neq 0\Rightarrow\mathcal{X}_v^l$ for every $v\in V$ and $l\in\{1,2,\dots,\vartheta\}$ (4)
  (if there is some agent in a vertex then the vertex is non-empty)
- $\mathcal{E}_{u,v}^l\Rightarrow\mathcal{X}_u^l\wedge\mathcal{X}_v^l$ for every $\{u,v\}\in E$ and $l\in\{1,2,\dots,\vartheta\}$ (5)
  $\mathcal{E}_v^l\Rightarrow\mathcal{X}_v^l\wedge\mathcal{X}_v^{l+1}$ for every $v\in V$ and $l\in\{1,2,\dots,\vartheta-1\}$ (6)
  (if an edge within a time layer or between time layers is non-empty then its both ends are non-empty)
- $\bigwedge_{u|\{u,v\}\in E}\neg\mathcal{E}_{u,v}^1$ for every $v\in V$ such that $(\exists a\in A)\alpha_0(a)=v$ (7)
  (for every source vertex at the first time layer all the incoming directed edges are empty)
  $\bigwedge_{v|\{u,v\}\in E}\neg\mathcal{E}_{u,v}^{\vartheta}$ for every $u\in V$ such that $(\exists a\in A)\alpha_+(a)=u$ (8)
  (for every destination vertex at the last time layer all the outgoing directed edges are empty)
- $\mathcal{E}_{u,v}^l\Rightarrow\mathcal{A}_u^l=\mathcal{A}_v^l$ for every $\{u,v\}\in E$ and $l\in\{1,2,\dots,\vartheta\}$ (9)
  $\mathcal{E}_v^l\Rightarrow\mathcal{A}_v^l=\mathcal{A}_v^{l+1}$ for every $v\in V$ and $l\in\{1,2,\dots,\vartheta-1\}$ (10)
  (if an edge is non-empty then there is the same agent at its both endpoints)
- $\mathcal{X}_u^l\Rightarrow\bigvee_{v|\{u,v\}\in E}\mathcal{E}_{u,v}^l\vee\mathcal{E}_u^l$ for every $u\in V$ and $l\in\{1,2,\dots,\vartheta-1\}$ (11)

$\sum_{v|\{u,v\}\in E} \mathcal{E}^l_{u,v} + \mathcal{E}^l_u \leq 1$ (12)
(if a vertex is non-empty at a time layer other than the last one then exactly one of its outgoing edges is non-empty as well)

$\mathcal{X}^\vartheta_u \Rightarrow \bigvee_{v|\{u,v\}\in E} \mathcal{E}^\vartheta_{u,v}$ for every $u \in V$ such that $(\forall a \in A)\alpha_+(a) \neq u$ (13)

$\sum_{v|\{u,v\}\in E} \mathcal{E}^\vartheta_{u,v} \leq 1$ (14)
(if a non-destination vertex at the last time layer is non-empty then exactly one of its outgoing edges is non-empty as well)

- $\mathcal{X}^l_v \Rightarrow \bigvee_{u|\{u,v\}\in E} \mathcal{E}^l_{u,v} \vee \mathcal{E}^{l-1}_v$ for every $v \in V$ and $l \in \{2,3,\dots,\vartheta\}$ (15)

$\sum_{u|\{u,v\}\in E} \mathcal{E}^l_{u,v} + \mathcal{E}^{l-1}_v \leq 1$ (16)
(if a vertex is non-empty at a time layer other than the first one then exactly one of its incoming edges is non-empty as well).

$\mathcal{X}^1_v \Rightarrow \bigvee_{u|\{u,v\}\in E} \mathcal{E}^1_{u,v}$ for every $v \in V$ such that $(\forall a \in A)\alpha_0(a) \neq v$ (17)

$\sum_{u|\{u,v\}\in E} \mathcal{E}^1_{u,v} \leq 1$ (18)
(if a non-source vertex at the first layer is non-empty then exactly one of its incoming edges is non-empty as well). □

Initial and goal arrangements are expressed as constraints over variables of the first and the last time layer. Note that some agents do not need to be assigned any goal if we do not care about their final positions.

The resulting formula of the *ϑ-RELAXED* encoding in the CNF form will be denoted as $F_{\vartheta-RE}(\Sigma)$. Without proof let us summarize the size of the encoding.

Proposition 1 ***(ϑ-RELAXED ENCODING SIZE).*** *The number of propositional variables in $F_{\vartheta-RE}(\Sigma)$ is $\mathcal{O}(\vartheta \cdot (|V| \cdot \lceil \log_2(\mu+1) \rceil + |E|))$ and the number of clauses is $\mathcal{O}(\vartheta \cdot ((|V|+|E|) \cdot \lceil \log_2(\mu+1) \rceil + |V|^3))$.* ■

A set $\Pi = \{\pi_1, \pi_2, \dots, \pi_\mu\}$ of vertex disjoint paths in $\mathrm{rExp_T}(\mathrm{G}, \vartheta)$ so that π_i connects $[\alpha_0(a_i),1]$ with $[\alpha_+(a_i), \vartheta]$ for $i = 1,2,\dots,\mu$ exists if and only if $F_{\vartheta-RE}(\Sigma)$ is satisfiable. The extraction of a solution of CPF Σ from a satisfying valuation of $F_{\vartheta-RE}(\Sigma)$ is shown using pseudo-code as Algorithm 2.

The algorithm tracks moves of agents towards their exits from the current time layer of the reduced time expansion graph during which the solution α is recorded. Note, that in each time layer the time step at which agents exit the layer is synchronized among all the agents (that is, agents exit at the same time step). It may therefore occur that agents wait for the last agent to finish its movements in the layer before they exit the layer together into the next one. The algorithm allows us to state the following theorem (proof is ommited).

Theorem 1 ***(SOLUTION OF Σ AND $F_{\vartheta-RE}(\Sigma)$ SATISFACTION).*** *A solution of a CPF $\Sigma = (G, A, \alpha_0, \alpha_+)$ with $A = \{a_1, a_2, \dots, a_\mu\}$ exists if and only if there exist $\vartheta \in \mathbb{N}$ for that formula $F_{\vartheta-RE}(\Sigma)$ is satisfiable.* ■

The original goal to reduce the size of the encoding by reducing the expansion of G is fulfilled by the fact that ϑ-RELAXED encoding needs no more time-layers than encodings for makespan optimal CPF solving. Moreover, there are cases where ϑ-RELAXED encoding needs significantly fewer time expansions – see example in Figure 2 where 3 time expansions are needed in ϑ-RELAXED encoding while makespan optimal encodings need 8 time expansions.

Algorithm 2. *Solution extraction algorithm for ϑ-RELAXED encoding.* A sequence of arrangements of agents forming a solution of given CPF Σ is extracted from satisfying valuation f of formula $F_{\vartheta-RE}(\Sigma)$ representing ϑ-RELAXED encoding of Σ.

input: Σ – an instance of CPF
ϑ – the number of time layers in ϑ-RELAXED encoding
f – a satisfying valuation of $F_{\vartheta-RE}(\Sigma)$
output: makespan and sequence of arrangements of agents forming the solution $\alpha_0, \alpha_1, \dots, \alpha_+$

function *Extract-Solution-ϑ-RELAXED* $(\Sigma = [G = (V,E), A, \alpha_0, \alpha_+], \vartheta, f)$: **pair**
1: $\eta_{\max} \leftarrow 0$ // time step at which movements at a time layer // are finished
2: **for each** $l = 1,2,\dots,\vartheta$ **do**
3: $\eta_{\min} \leftarrow \eta_{\max}$ // time step at which movements // at a time layer start
4: **for each** $a \in A$ **do**
5: $\eta \leftarrow \eta_{\min}$
6: $u \leftarrow \alpha_{\eta_{\min}}(a)$
7: **while** $(l \neq \vartheta$ **and** $f(\mathcal{E}^l_u) = FALSE)$ **or** $(l = \vartheta$ **and** $u \neq \alpha_+(a))$ **do**
8: $\alpha_{\geq\eta}(a) \leftarrow u$ // agent a will be located in u // at all the time steps $\geq \eta$
9: **for each** $v \in V$ such that $\{u,v\} \in E$ **do**
10: **if** $f(\mathcal{E}^l_{u,v}) = TRUE$ **then**
11: $u \leftarrow v$
12: $\eta \leftarrow \eta + 1$
13: $\eta_{\max} \leftarrow \max(\eta_{\max}, \eta)$
14: $\alpha_{\geq\eta}(a) \leftarrow u$
15: **return** $(\eta_{\max}, [\alpha_0, \alpha_1, \dots, \alpha_{\eta_{\max}}])$

Proposition 2 ***(ADVANTAGE OF ϑ-RELAXED ENCODING).*** *Let η be an optimal makespan achievable in a CPF Σ. Then $F_{\vartheta-RE}(\Sigma)$ is solvable for $\vartheta \leq \eta$. Moreover, there exists a CPF instance Σ where strict inequality $\vartheta < \eta$ holds.* ■

The number of time layers in ϑ-RELAXED encoding that grants finding a solution corresponds rather to the intensity of interactions among agents. Hence to further reduce the size of the encoding via reducing the number of time layers we suggest decomposing solving of a given CPF Σ into solving multiple CPFs in which intesity of interactions among agents is low and thus they can be solved by satifying ϑ-RELAXED encoding formualae consisting of few time layers.

The suggested decomposition corresponds to placing agents to their goals one by one while individual CPFs represents relocating a single agent where positions of previously placed agents are preserved. The process is called UniAGENT solving and it is formally described as Algorithm 3.

Without proof let us state that the UniAGENT method is **sound**; that is, it always finds a solution provided a solution

exists. This is due to the fact, that we do constrain only agents that have been placed so far while remaining agents can be placed arbitrarily. This in theory tells that all the sub-goals determined by single agent placement are fesible.

Algorithm 3. *UniAGENT SAT-based CPF solving.* Agents (robots) are placed to their goals one by one. Relocation of a single agent to its goal is solved as an individual CPF using ϑ-RELAXED encoding where already placed agents preserve their positions. Relatively small difference between the initial arrangement and goal in single agent relocation CPFs allows to solve them with few time layers in the reduced time expansion graph.

input: Σ – an instance of CPF
output: makespan and a sequence of arrangements of agents of arrangements of agents forming the solution

function *Solve-UniAGENT* ($\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$): **pair**
let $A = \{a_1, a_2, \dots, a_\mu\}$
$\eta_{\max} \leftarrow 0$
for each $i = 1,2, \dots, \mu$ **do**
  $\beta_0 \leftarrow \alpha_{\eta_{\max}}$
  for each $j = 1,2, \dots, i-1$ **do**
    $\beta_+(a_j) \leftarrow \alpha_{\eta_{\max}}(a_j)$
  $\beta_+(a_i) \leftarrow \alpha_+(a_i)$
  $(\vartheta, f) \leftarrow$ *Find-Optimal-Parameter* $(\Phi = [G, A, \beta_0, \beta_+])$
  $(\eta, \vec{s}) \leftarrow$ *Extract-Solution-ϑ-RELAXED* (Φ, ϑ, f)
  for each $k = 0,1, \dots, \eta - 1$ **do**
    $\alpha_{\eta_{\max}+k} \leftarrow \vec{s}[k]$
  $\eta_{\max} \leftarrow \eta_{\max} + \eta$
return $(\eta_{\max}, [\alpha_0, \alpha_1, \dots, \alpha_{\eta_{\max}}])$

In our minor experiments, we found that the ϑ-RELAXED encoding is very easy to solve if there are few time layers but it gets rapidly harder with the increasing number of time layers. The number of time layers necesssary to reach the solvability when a single agent is relocated is typically very low (usually 1 to 3 time layers). Moreover, the makespan of solutions generated by the UniAGENT solving process is similar to that of generated by solving the ϑ-RELAXED encoding where all the agents are relocated at once in cases where we managed to solve the ϑ-RELAXED encoding. These observations together justifies the use of the new encoding as suggested in the UniAGENT solving process.

5 Experimental Evaluation

Series of experiments have been conducted in order to evaluate the suggested propositional ϑ-RELAXED encoding and UniAGENT solving process based on it.

The comparison has been done with existent encodings for makespan optimal CPF solving – INVERSE, ALL-DIFFERENT, DIRECT, MATCHING, and SIMPLIFIED [Surynek, 2012a, 2012b, 2014]. To include other than SAT-based methods, the comparison with A*-based OD+ID [Standley and Korf, 2011] for makespan optimal solving is also presented. Makespan suboptimal methods are represented by WHCA* [Silver, 2005] in our comparison.

We used benchmarks suggested in [Silver, 2005] which consist of randomly generated CPF instances over **4-connected grids** with randomly placed obstacles. There are also randomly placed **obstacles** by which 20% of all the vertices are occupied. An important module in the whole solving process is a SAT solver. `Glucose` version `3.0` [Audemard and Simon, 2013] has been used in the experimental evaluation.

5.1 Encoding Size Comparison

The important characteristic of propositional formulae with respect to the speed of their solving is their size while small is preferable (the size is represented by the *number of variables and clauses* in our case).

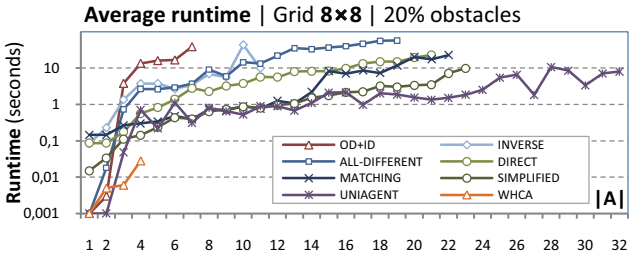
Selected results are shown in Table 1. Size measurement is done on 4-connected grid and for various numbers of agents in the environment. For each number of agents **10 random instances** were generated and average value for each characteristic is presented.

Table 1. ***Size comparison*** *of encodings over 8×8 grid.* INVERSE, ALL-DIFFERENT, DIRECT, MATCHING, SIMPLIFIED [Surynek, 2012a, 2012b, 2014] and ϑ-RELAXED encodings are compared. CPF instances are generated over the 4-connected grid of size 8×8 with 20% of cells occupied by obstacles. Makespan bound η and the number of time layers in reduced time expansion graph ϑ is always 16. The number of *variables* and *clauses*, the *ratio* of the number of clauses and the number of variables, and the *average clause length* are listed for different sizes of the of agents A. The advantage of ϑ-RELAXED encoding is that it is relatively small compared to other encodings.

Grid 8×8 |Agents|			INVERSE		ALL-DIFFERENT		DIRECT		MATCHING		SIMPLIFIED		θ-RELAXED	
1	#Variables	*Ratio*	8 358.7	*3.748*	1 489.3	*5.325*	**814.4**	*28.539*	4 520.3	5.710	1 628.8	*2.078*	4 645.1	*4.358*
	#Clauses	*Length*	31 327.9	*2.616*	7 930.4	*3.057*	23 241.9	***2.149***	25 881.1	2.441	**3 384.6**	*2.550*	20 246.6	*2.515*
4			10 019.5	*5.532*	7 834.5	*4.440*	**3 257.6**	*35.589*	6 181.1	6.984	4 072.0	*4.420*	6 273.9	*5.404*
			55 437.0	*2.641*	34 781.9	*3.103*	115 934.3	***2.272***	43 171.0	2.640	**17 997.8**	*2.374*	33 904.1	*2.660*
16			11 680.3	*7.820*	67 088.3	*3.231*	13 030.4	*64.506*	**7 841.9**	9.215	13 844.8	*10.853*	7 902.7	*5.988*
			91 344.5	*3.127*	216 745.4	*3.147*	840 540.6	*2.505*	72 259.3	3.315	150 259.2	***2.180***	**47 324.6**	*2.714*
32			12 510.7	*9.765*	230 753.0	*2.802*	26 060.8	*105.084*	**8 672.3**	11.494	26 875.2	*19.002*	8 717.1	*6.159*
			122 170.3	*3.733*	646 616.2	*3.168*	2 738 584.7	*2.621*	99 675.5	4.045	510 672.1	***2.111***	**53 697.0**	*2.722*

It can be observed from presented results that the ϑ-RELAXED encoding is the **s**mallest in terms of the number of clauses and the second smallest in terms of the number of variables just after the MATCHING encoding. The average clause length also indicates that most of clauses are binary.

Note, that formulae for all the encodings were generated with the same number of time-layers. In most cases however, ϑ-RELAXED encoding needs fewer time-layers to achieve solvability.

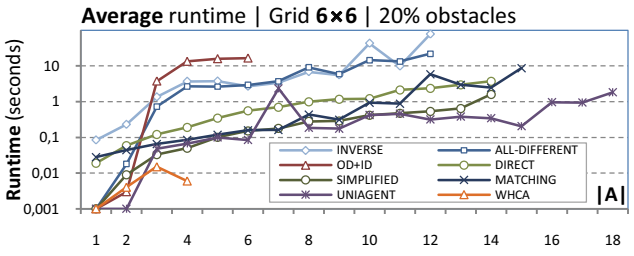


|A|	1	4	8	12	16	20	24	28	32
η	5.3	8.4	11.0	11.7	12.4	12.3	-	-	-
ω	5.6	9.3	-	-	-	-	-	-	-
ϑ	9.3	15.8	33.0	49.3	83.4	96.1	131.4	154.1	201.7

Figure 3. *Runtime and makespan comparison over 8×8 grid.* UniAGENT and WHCA* produce makespan sub-optimal solutions; all other methods are makespan optimal. Evaluation of runtime and makespan was done for the growing number of agents (timeout is 256 seconds). Average optimal makespan is shown as η; ϑ and ω are average makespans of UniAGENT and WHCA* respectively.

5.2 Runtime Evaluation

Runtime tests were done over 4-connected grids with growing number of agents. The *timeout* has been set to **256 seconds** and for each number of agents **10 random instances** were solved while runtime was recorded – average runtime is presented.



|A|	1	2	4	6	8	12	14	16	18
η	4.2	4.9	5.6	7.0	7.4	7.9	8.6	-	-
ω	4.3	5.3	5.7	-	-	-	-	-	-
ϑ	5.7	8.5	11.1	16.7	30.2	43.1	49.3	50.5	87.3

Figure 4. *Runtime and makespan comparison over 6×6 grid.* The UniAGENT solving is almost by order of magnitude faster than second best method for higher number of agents.

Runtime results are presented in Figure 3 and Figure 4. Average optimal makespan and average sub-optimal makespan obtained with the UniAGENT and WHCA* methods are also shown. It can be observed that OD+ID and WHCA* although performing as best for small number of agents, quickly reaches the timeout as the number of agents grows. UniAGENT method scales up as the best for growing number of agents though the makespan is up to several times longer than the optimum. Up to 30 agents (occupancy 83%) and up to 48 agents (occupancy of 75%) can be solved in 6×6 and 8×8 grid respectively with no obstacles within the timeout of 1.0 minute.

Motivated by experiments presented in [Standley and Korf, 2011], we also tried to solve (N^2-2)-puzzles by the UniAGENT solver; that is, 4-connected grids with two blanks (two blanks grant that instances are solvable). In these situations, A*-based solvers relying on independence detection such as OD+ID and MGS1 do not scale well. The (3^2-2)-puzzles were solved in less than 1.0 second by UniAGENT solver. The (4^2-2)-puzzles needed approximately 10 seconds. Larger puzzles have not been solved under 1.0 minute.

Table 2. *Makespan comparison with* ***domain independent planners***. Suboptimal planners LPG-td and SGPLAN managed to solve instances over the 6×6 grid with 20% obstacles with up to 6 agents within the timeout of 256 seconds. UNIAGENT solver generates solutions of shorter makespan and is much faster.

|A| in 6×6	1	2	3	4	5	6	7
LPG-td	17.2	7.6	18.5	16.2	22.7	134.1	-
SGPLAN	7.2	9.8	16.7	15.1	23.4	-	-
UNIAGENT	5.7	8.5	12.3	11.1	15.9	16.7	20.5

We also made comparison with several domain independent planners including SAT-based makespan optimal SATPLAN [Kautz and Selman, 1999] and SASE [Huang *et al.*, 2010] and makespan suboptimal LPG-td [Gerevini *et al.*, 2008] and SGPLAN [Hsu *et al.*, 2006]. Planners were run on instances over 6×6 grid with 20% obstacles containing few agents - part of results is shown in Table 2. SATPLAN and SASE performed orders of magnitude worse than SAT-based solving with refered domain dependent encodings (thus not presented). Makespan suboptimal planners LPG-td and SGPLAN performed much better but still do not scale up. Moreover, they tend to generate worse makespans than the UniAGENT method.

6 Conclusions

The concept of *reduced time expansion* graph and ϑ-RELAXED propositional encoding of CPF based on this graph have been introduced. The search for a solution of CPF is reduced to the search of **vertex disjoint paths** in reduced time expansion graph which is done via SAT solving. In order to maximally reduce the size of the propositional encoding, the search for a goal arrangement is decomposed into multiple searches for sub-goals which correspond to placement of a single agent.

Experimental evaluation indicates that the novel CPF solving method - called UniAGENT solver - **scales up** better for higher number of agents than comparable makespan suboptimal seach-based method WHCA*. The relaxation from the requirement on the makespan optimality allowed significant **runtime improvement** compared to other propositional encodings and related SAT-based solving schemes. This advanced applicability of SAT-based CPF solving in **higly constrained** situations towards even denser occupacy with agents.

Although solutions generated by the UniAGENT method are makespan suboptimal, they are obtained through optimization of a diffrent parameter - namely the number of time layers in the ϑ-RELAXED encoding - hence their makespan is not as dramatically far from the optimum as in the case of rule based algorithms like PUSH-AND-ROTATE [de Wilde et al., 2014]. Altogether, UniAGENT solver represents a viable alternative to existing rule and search based CPF solvers.

References

[Ahuja *et al.*, 1993] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. *Network flows: theory, algorithms, and applications.* Prentice Hall, 1993.

[Audemard and Simon, 2013] Audemard, G., Simon, L. *The Glucose SAT Solver.* http://labri.fr/perso/lsimon/glucose/, 2013.

[Barer et al., 2014] Barer, M., Sharon, G., Stern, R., Felner, A. *Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem.* ECAI 2014 - 21st European Conference on Artificial Intelligence (ECAI 2014), pp. 961-962, IOS Press, 2014.

[Biere *et al.*, 2009] Biere, A., Heule, M., van Maaren, H., Walsh, T. *Handbook of Satisfiability.* IOS Press, 2009.

[Čáp *et al.*, 2013] Čáp, M., Novák, P., Vokřínek, J., Pěchouček, M. *Multi-agent RRT: sampling-based cooperative pathfinding.* International conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013), pp. 1263-1264, IFAAMAS, 2013.

[Erdem *et al.*, 2013] Erdem, E., Kisa, D. G., Öztok, U., Schüller, P. *A General Formal Framework for Pathfinding Problems with Multiple Agents.* Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013), AAAI Press, 2013.

[Gerevini *et al.*, 2008] Gerevini, A., Saetti, A., Serina, I., Toninelli, P. *LPG-td: fully automated planner for PDDL2.2 domains*, research web page, University of Brescia, Italy, http://zeus.ing.unibs.it/ lpg, 2008.

[Hsu *et al.*, 2006] Hsu, C. W., Wah, B. W., Huang, R., Chen, Y. X. *SGPlan 5: Subgoal Partitioning and Resolution in Planning*, research web page, University of Illinois, USA, http://wah.cse.cuhk.edu.hk/wah/programs/SGPlan, 2006.

[Huang *et al.*, 2010] Huang, R., Chen, Y., Zhang, W. *A Novel Transition Based Encoding Scheme for Planning as Satisfiability.* Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), AAAI Press, 2010.

[Kautz and Selman, 1999] Kautz, H., Selman, B. *Unifying SAT-based and Graph-based Planning.* Proceedings of the 16th International Joint Conference on Artificial Intelligence, pp. 318-325, Morgan Kaufmann, 1999.

[Kornhauser *et al.*, 1984] Kornhauser, D., Miller, G. L., Spirakis, P. G. *Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications.* Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 241-250, IEEE, 1984.

[Ratner and Warmuth, 1986] Ratner, D., Warmuth, M. K. *Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable.* Proceedings of the 5th National Conference on Artificial Intelligence (AAAI 1986), pp. 168-172, Morgan Kaufmann, 1986.

[Ryan, 2008] Ryan, M. R. K. *Exploiting Subgraph Structure in Multi-Robot Path Planning.* Journal of Artificial Intelligence Research, Volume 31, pp. 497-542, AAA Press, 2008.

[Sharon et al., 2013] Sharon, G., Stern, R., Goldenberg, M., Felner, A. *The increasing cost tree search for optimal multi-agent pathfinding.* Artificial Intelligence, Volume 195, pp. 470-495, Elsevier, 2013.

[Silver, 2005] Silver, D. *Cooperative Pathfinding.* Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005), pp. 117-122, AAAI Press, 2005.

[Standley and Korf, 2011] Standley, T. S., Korf, R. E. *Complete Algorithms for Cooperative Pathfinding Problems.* Proceedings of Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 668-673, IJCAI/AAAI Press, 2011.

[Surynek, 2012a] Surynek, P. *Towards Optimal Cooperative Path Planning in Hard Setups through Satisfiability Solving*. Proceedings of 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2012), pp. 564-576, Springer, 2012.

[Surynek, 2012b] Surynek, P. *On Propositional Encodings of Cooperative Path-Finding*. Proceedings of the 24th International Conference on Tools with Artificial Intelligence (ICTAI 2012), pp. 524-531, IEEE, 2012.

[Surynek, 2013] Surynek, P. *Mutex reasoning in cooperative path finding modeled as propositional satisfiability.* Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013), pp. 4326-4331, IEEE, 2013.

[Surynek, 2014] Surynek, P. *Simple Direct Propositional Encoding of Cooperative Path Finding Simplified Yet More.* Proceedings of the 13th Mexican International Conference on Artificial Intelligence (MICAI 2014), pp. 410-425, Springer, 2014.

[Wagner and Choset, 2015] Wagner, G., Choset, H. *Subdimensional expansion for multirobot path planning.* Artificial Intelligence, Volume 219, pp. 1-24, Elsevier, 2015.

[Wehrle and Rintanen, 2007] Wehrle, M., Rintanen, J. *Planning as satisfiability with relaxed exist-step plans.* Proceedings of the 20th Australian Joint Conference on Artificial Intelligence, pp. 244–253, Springer, 2007.

[de Wilde et al., 2014] de Wilde, B., ter Mors, A. W., Witteveen, C. *Push and Rotate: a Complete Multi-agent Pathfinding Algorithm.* Journal of Artificial Intelligence Research, Volume 51, pp. 443-492, AAAI Press, 2014.

[Yu and LaValle, 2013] Yu, J., LaValle, S. M. *Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs.* Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013), AAAI Press, 2013.