Compact Representations of Cooperative Path-Finding as SAT Based on Matchings in Bipartite Graphs

Pavel Surynek
Charles University Prague, Malostranské náměstí 25, Praha, 118 00, Czech Republic
pavel.surynek@mff.cuni.cz

***Abstract*— This paper addresses makespan optimal solving of cooperative path-finding problem (CPF) by translating it to propositional satisfiability (SAT). The task is to relocate set of agents to given goal positions so that they do not collide with each other. A novel SAT encoding of CPF is suggested. The novel encoding uses the concept of matching in a bipartite graph to separate spatial constraint of CPF from consideration of individual agents. The separation allowed reducing the size of encoding significantly. The conducted experimental evaluation shown that novel encoding can be solved faster than existing encodings for CPF and also that the SAT based methods dominates over A* based methods in environment densely occupied by agents.**

Keywords-cooperative path-finding (CPF); propositional satisfiability (SAT); encodings; matchings; bipartite graphs; A*

I. Introduction and Context

The problem of *cooperative path-finding* (CPF) [13], [17], [19], [23] represents an abstraction for variety of problems where the task is to relocate some physical agents, robots, or other objects so that they do not collide with each other. Each agent is given its initial position in a certain environment and its task is to reach a given goal position. It is assumed that all the agents are the same (same size and velocity) and are controlled centrally. That is, agents make no decisions themselves (although sometimes the problem is referred to as *multi-agent path-finding*, it is actually not a multi-agent system). The centralized planning mechanism finds a spatial-temporal path for each agent through which the agent can relocate to its goal. The major difficulty in CPF comes from possible interactions among relocated agents, which is imposed by the requirement that they must not collide with each other. The more agents appear in the instance the more complex interaction arises and consequently the instance is harder to solve.

There are many **motivations** for introducing CPF. Classical multi-robot relocation problems where agents are represented by actual mobile robots can be viewed as CPF. Planning movements of units in real-time strategy games is another application [23]. Even data relocation in a network can be regarded a CPF (agent is represented by a data packet and spatial occupancy turns into storage occupancy).

This research is work is supported by the Czech Science Foundation under the contract number GAP103/10/1287.

The indifference between agents in terms of their properties allows abstraction where the environment is modeled as an undirected graph and agents as items placed in vertices of this graph [19], [23]. Spatial properties of agents are modeled by the requirement that at most one agent is placed in each vertex. The time is discrete and the move is possible only into a currently unoccupied vertex while no other agent is allowed to enter the same target vertex. It can be observed now, that well-known relocation puzzles such as N×N-puzzle [12], [14]are special cases of CPF.

Contemporary approaches to solving CPF include polynomial time sub-optimal algorithms [13], [24] as well as methods that generate optimal solutions in certain sense [20], [21]. This work focuses on generating *makespan optimal* solutions to CPF where the makespan is the maximum of arrive times over all the agents.

Related makespan optimal methods for CPF currently include methods employing translation of CPF to *propositional satisfiability* (SAT) [22], methods based on *conflict resolution* between paths for individual agents [18], and classical *A* based methods* equipped with powerful heuristics [20]. The first mentioned approach excels in relatively small environments with high density of agents while latter two approaches are better in large environments with few agents and low interaction among them.

This work tries to contribute to SAT-based methods. Particularly, a novel propositional encoding of CPF is introduced in this paper. The new encoding is based on the concept of *matching* in a *bipartite graph*. It is smaller and can be solved faster than previous two encodings. It is also shown how the SAT-based solving stands in comparison with A* based methods.

The **organization** of the paper is as follows. The CPF problem is introduced formally first. Then the novel propositional encoding of CPF is described and its theoretical properties are summarized. Experimental evaluation in which two previous encodings and the A* based method are compared with the novel encoding constitute the last part.

II. Cooperative Path-Finding (CPF) Formally

An arbitrary **undirected graph** can be used to model the environment where agents are moving. Let $G = (V, E)$ be such a graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set of ver-

tices and $E \subseteq \binom{V}{2}$ is a set of edges. The placement of agents in the environment is modeled by assigning them vertices of the graph. Let $A = \{a_1, a_2, \dots, a_\mu\}$ be a finite set of *agents*. Then, an arrangement of agents in vertices of graph G will be fully described by a *location* function $\alpha: A \rightarrow V$; the interpretation is that an agent $a \in A$ is located in a vertex $\alpha(a)$. At most **one agent** can be located in each vertex; that is α is uniquely invertible. A generalized inverse of α denoted as $\alpha^{-1}: V \rightarrow A \cup \{\perp\}$ will provide us an agent located in a given vertex or $\perp$ if the vertex is empty.

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. □

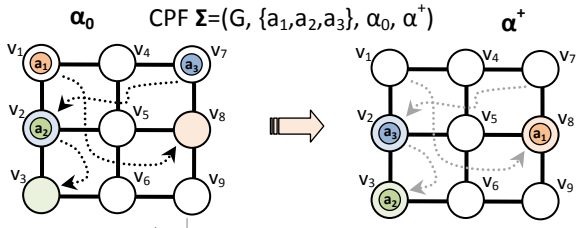
The dynamicity of the model supposes a discrete time divided into time steps. An arrangement α_i at the i-th time step can be transformed by a transition action which instantaneously moves agents in the non-colliding way to form a new arrangement α_{i+1}. The resulting arrangement α_{i+1} must satisfy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

(i) $\forall a \in A$ either $\alpha_i(a) = \alpha_{i+1}(a)$ or $\{\alpha_i(a), \alpha_{i+1}(a)\} \in E$ holds (agents move along edges or not move at all), (1)

(ii) $\forall a \in A \;\; \alpha_i(a) \neq \alpha_{i+1}(a) \Rightarrow \alpha_i^{-1}(\alpha_{i+1}(a)) = \perp$ (agents move to vacant vertices only), and (2)

(iii) $\forall a, b \in A \;\; a \neq b \Rightarrow \alpha_{i+1}(a) \neq \alpha_{i+1}(b)$ (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 Fig. 1.



$\vec{s}$	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

Fig. 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.

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 $\vec{s} = [\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_m]$ where $\alpha_m = \alpha^+$ and α_{i+1} is a result of valid transformation of α_i for every $i = 1,2, \dots, m-1$. □

A notation $|\vec{s}|$ will be also used to denote the makespan. If it is a question whether there exists a solution of Σ of the makespan at most a given bound η we are speaking about a *bounded CPF (bCPF)*. It is known that bCPF is NP-complete and finding makespan optimal solution to CPF is NP-hard [14]. Notice that due to no-ops introduced in valid transitions it is equivalent to finding a solution of the makespan equal to the given bound.

III. A NOVEL ENCODING BASED ON BIPARTITE MATCHING

The core idea behind the suggested encoding is to divide requirement on occupancy of vertices by at most one agent and collision avoidance constraints from mapping of agents to individual vertices. If a movement takes place then the target vertex must be vacant and no other agent is allowed to enter the target at the same time. Such a requirement can be modeled even if it is not know what particular agent moves. In other words, occupancy and collision requirements can be tested with *anonymous* agents.

The mapping of individual agents to vertices is done separately. Locations of an actual (not anonymous) agent can be mapped to locations of anonymous one provided, that the same (actual) agent is mapped to source and target locations of each move of anonymous agent. The advantage of separating occupancy and collision constraints and agent mapping is that the former does not need to care about the latter, which significantly simplifies expression of associated constraints by means of propositional satisfiability. Occupancy and collision constrains with anonymous agents do not need to distinguish between individual agents while agents do not need observe locations of other agents.

The cooperation of both views of the problem to ensure the model is a correct model of solution existence in bCPF is carried out through *channeling constraints* [6], [7]. The concept of anonymous agents can be regarded as anonymous material that is transported around graph.

The closest concept from computer science that captures such anonymous material transportation is the concept of *network flow* [1]. Without stating more details, let us note that if network flows are employed in solving CPF, then capacities of edges of a graph modeling the instance should be 1, which expresses the fact that agents should not collide and must not share any vertex. If a flow is finally extracted from such a graph and sliced into layers corresponding to individual time steps then saturated edges in such slices will form *matchings* and slices itself *bipartite graphs* [1],[15]. The bipartite graph consists of vertices visited by material (agents) in consecutive time steps and of edges representing possible moves.

However, the situation is not that straightforward. It cannot be modeled by a matching in a bipartite graph that an agent always moves into a vacant vertex. Hence, additional treatment is needed.

A. Background of the Encoding

Let us have an instance of bCPF $\Sigma = [G, A, \alpha_0, \alpha_+]$, $G = (V, E)$ with makespan bound η. A *time expansion undi-*

rected graph G_η is intended to represent process of agent relocation over the original graph G in time. $G_\eta = (V_\eta, E_\eta)$ will be defined as follows: $V_\eta = V \times \{0,1,\dots,\eta\}$, that is, we have a copy of the original set of vertices for each time step. A set of vertices $V \times \{i\}$ for $i \in \{0,1,\dots,\eta\}$ will be called an i-th *layer*. Edges are introduced as follows: $\{[u,i];[u,i+1]\} \in E_\eta$ for every $u \in V$ and $\{[u,i];[v,i+1]\} \in E_\eta$ for every $(u,v) \in E$ for $i \in \{0,1,\dots,\eta-1\}$. A movement of agents between time steps i and $i+1$ for $i \in \{0,1,\dots,\eta-1\}$ represented by location functions α_i and α_{i+1} respectively can be recorded by assigning agents to the set of vertices $V \times \{i\}$ and $V \times \{i+1\}$ according to α_i and α_{i+1} respectively. An *induced matching* with respect to a movement defined by α_i and α_{i+1} in bipartite graph $G_\eta|_{V\times\{i,i+1\}}$ is obtained by taking edges having the same assigned agents at both ends (see Fig. 2 for illustration).

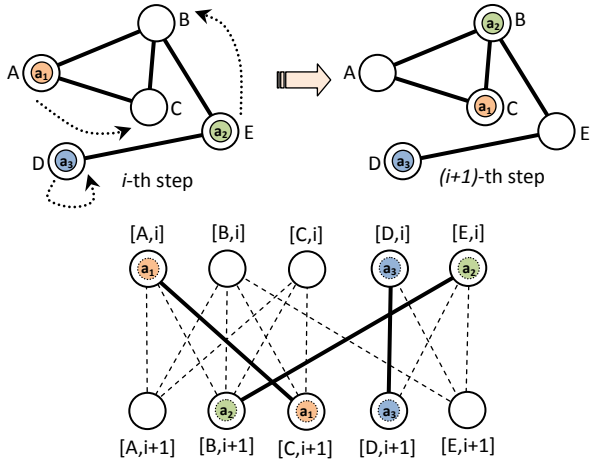


Fig. 2. ***Correspondence*** *of agent movement and matching in bipartite graph.* Movement between time steps i and $i+1$ and corresponding induced matching are illustrated. A set of source vertices of non-trivial moves and a set of their targets must be disjoint in a matching that corresponds to a valid movement ($\{A,E\} \cap \{B,C\} = \emptyset$).

Proposition 1 (INDUCED MATCHINGS). *The set of induced matchings between time steps in the time expansion graph forms a proper subset of all matchings.* ■

Proof. Validity constraints in the definition of valid movement (ii) and (iii) require that the set of source vertices of non-trivial moves (that is, moves in which the agent does not stay in a vertex) and the set of their targets are disjoint. That is, $\{\alpha_i(a) | a \in A \wedge \alpha_i(a) \neq \alpha_{i+1}(a)\} \cap \{\alpha_{i+1}(a) | a \in A \wedge \alpha_i(a) \neq \alpha_{i+1}(a)\} = \emptyset$ (see Fig. 2 for illustration). Obviously, not all matchings satisfy this condition (see Fig. 3). ■

Thus, if motions between time steps in CPF are modeled as matchings in time expansion graphs it is necessary to rule out matchings that are not induced. One can see that ruling out such matching can be done by adding constraints that enforce empty intersection between sources and targets of non-trivial moves. Although posting such constraints on searched matchings while preserving completeness is difficult in matching search algorithms such as in those based on network flows, the situation in SAT is much more promising thanks to its expressive power.

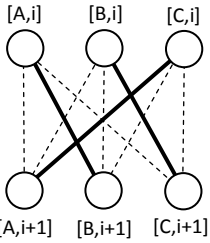


Fig. 3. *A matching that is* ***not induced*** *by movement.* The set of source and target vertices of non-trivial moves have non-empty intersection ($\{A,B,C\} \cap \{A,B,C\} \neq \emptyset$).

Induced matchings can be connected together into a sequence along the whole time expansion graph, that is from 0-th layer to η-th layer. Connecting matchings in sequence means to connect target vertices of one matching with source vertices of the successor matching (see Fig. 4). If initial locations of agents in the 0-th layer are connected with goals in the η-th layer of the time expansion graph through sequence of induced matchings then it forms a reservation in time and space through which valid movements of agents from initial locations to their goals can be done.

However, the concept of matching does not allow mapping of agent's initial locations to their goals on an individual basis (agents may be arbitrarily permuted in their goals if regarded through induced matchings only). Therefore, further additional constraints are necessary. These will however not be included in the matching model, which we would like to keep as simple as possible. A separate model, which will map agents to individual vertices, is introduced.

B. Details of the Encoding

The suggested SAT encoding – called MATCHING encoding – is divided into two parts. The first one models occupancy/collision requirements and it is based on induced matchings in time expansion graph. The second part models mapping of agents to vertices but does not care about collision and occupancy requirements.

Definition 3 (INDUCED SUBMODEL). An *induced part* of the MATCHING encoding of given bCPF consists of propositional variable for each vertex and edge in the time expansion graph. That is, propositional variable $\mathcal{M}_v^i$ is introduced for every $i = 0,1,\dots,\eta$ and $v \in V$ and propositional variables $\mathcal{E}_{u,v}^i$ and $\mathcal{E}_u^i$ are introduced for every $i = 0,1,\dots,\eta$ and $\{u,v\} \in E$ and $u \in V$ respectively. Constraints enforce that variables set to $TRUE$ form an induced matching:

(a) $\mathcal{E}_{u,v}^i \Rightarrow \mathcal{M}_u^i \wedge \mathcal{M}_v^{i+1}$ for every $\{u,v\} \in E$ and $i \in \{0,1,\dots,\eta-1\}$, (4)

$\mathcal{E}_u^i \Rightarrow \mathcal{M}_u^i \wedge \mathcal{M}_u^{i+1}$ for every $u \in V$ and $i \in \{0,1,\dots,\eta-1\}$ (5)

(if an edge is selected into matching then its endpoints are selected as well)

(b) $\mathcal{E}_u^i + \sum_{v|\{u,v\}\in E} \mathcal{E}_{u,v}^i \leq 1$ for every $u \in V$ and $i \in \{0,1,\dots,\eta-1\}$, (6)

$\mathcal{E}_v^i + \sum_{u|\{u,v\}\in E} \mathcal{E}_{u,v}^i \leq 1$ for every $v \in V$ and $i \in \{0,1,\dots,\eta-1\}$, (7)

(at most one incoming and outgoing edge is selected into matching)

(c) $\mathcal{M}_u^i \Rightarrow \mathcal{E}_u^i \vee \bigvee_{v|\{u,v\}\in E} \mathcal{E}_{u,v}^i$ for every $u \in V$ and $i \in \{0,1,\dots,\eta-1\}$, (8)

$\mathcal{M}_v^{i+1} \Rightarrow \mathcal{E}_u^i \vee \bigvee_{u|\{u,v\}\in E} \mathcal{E}_{u,v}^i$ for every $v \in V$ and $i \in \{0,1,\dots,\eta-1\}$, (9)

(if a vertex is selected into matching then at least one outgoing and incoming edge must be selected as well)

(d) $\mathcal{E}_{u,v}^i \Rightarrow \neg\mathcal{M}_v^i$ for every $\{u,v\} \in E$ and $i \in \{0,1,\dots,\eta-1\}$, (10)

(source and target vertices of non-trivial moves must be disjoint). □

$\mathcal{M}_v^i$ indicates that vertex (v,i) has been selected into matching in the time expansion graph. Similarly, $\mathcal{E}_{u,v}^i$ and $\mathcal{E}_u^i$ indicate that edge $\{(u,i),(v,i+1)\}$ and $\{(u,i),(u,i+1)\}$ respectively have been selected into matching. Block of constraints (a) ensures consistency between selection of vertices and edges; constraints (b) and (c) ensures Kirchhoff's laws [1] on selection of edges, and finally (d) constraints ensures that the selected matching is induced one.

For simplicity, the mapping part of the encoding will be introduced over finite domain integer variables, which will be subsequently substituted with bit vectors while constraints will be translated into clauses.

Definition 4 (MAPPING SUBMODEL). A *mapping part* of the MATCHING encoding of a given bCPF consists of variables $\mathcal{A}_v^i \in \{0,1,\dots,|A|\}$ for every $i = 0,1,\dots,\eta$ and $v \in V$. Constraints will form a channel to the induced part of the model:

$\mathcal{E}_{u,v}^i \Rightarrow \mathcal{A}_u^i = \mathcal{A}_v^{i+1}$ for $\{u,v\} \in E$ and $i \in \{0,1,\dots,\eta\}$ (11)

$\mathcal{A}_u^i \neq 0 \Rightarrow \mathcal{M}_u^i$ for $u \in V$ and $i \in \{0,1,\dots,\eta\}$. □ (12)

Variable $\mathcal{A}_v^i$ indicate what agent appear in vertex (v,i) (equivalently what agent is in v at time step i), where $\mathcal{A}_v^i = 0$ means that (v,i) is vacant. Constraints merely say that if an edge is selected in to matching, then an agent must be transported along it (constraints (e)). Whenever there is some agent in a vertex then this vertex must be selected into matching (constraints (f)).

Initial and goal arrangement will be expressed through additional constraints on variables corresponding to 0-th and η-th layer of the time expansion graph:

Initial: $\begin{cases} \mathcal{A}_{v_i}^0 = j \wedge \mathcal{M}_{v_i}^0 & \text{iff} \quad \alpha_0^{-1}(v_i) = a_j \\ \mathcal{A}_{v_i}^0 = 0 \wedge \neg\mathcal{M}_{v_i}^0 & \text{iff} \quad \alpha_0^{-1}(v_i) = \perp \end{cases}$ (13)

Goal: $\begin{cases} \mathcal{A}_{v_i}^\eta = j \wedge \mathcal{M}_{v_i}^\eta & \text{iff} \quad \alpha_+^{-1}(v_i) = a_j \\ \mathcal{A}_{v_i}^\eta = 0 \wedge \neg\mathcal{M}_{v_i}^\eta & \text{iff} \quad \alpha_+^{-1}(v_i) = \perp \end{cases}$ (14)

It remains to show how to translate constraints (6) and (7) to conjunction of clauses and how to deal with integer variables $\mathcal{A}_v^i$. Constraint (6), that is, $\mathcal{E}_u^i + \sum_{v|\{u,v\}\in E} \mathcal{E}_{u,v}^i \leq 1$ will be substituted by differences between all the pairs of variables:

$$\bigwedge_{v,w|\{u,v\}\in E\wedge\{u,w\}\in E\wedge v\neq w} \neg\mathcal{E}_{u,v}^i \vee \neg\mathcal{E}_{u,w}^i \qquad (15)$$
$$\bigwedge_{v|\{u,v\}\in E} \neg\mathcal{E}_{u,v}^i \vee \neg\mathcal{E}_u^i$$

Constraint (7) is treated analogically. Finite domain integer variables $\mathcal{A}_v^i$ will be represented by vectors of $\lceil \log_2(|A|+1) \rceil$ propositional variables using *binary encoding*. Individual propositional variables representing $\mathcal{A}_v^i$ will be accessed by indexing: $\mathcal{A}_v^i[\mathbb{i}]$ is $\mathbb{i}$-th propositional variable representing $\mathcal{A}_v^i$. Then constraints (11) and (12) in mapping part of the model can be respectively translated as follows:

$$\mathcal{E}_{u,v}^i \Rightarrow \bigwedge_{\mathbb{i}=1}^{\lceil \log_2(|A|+1) \rceil} (\neg\mathcal{A}_u^i[\mathbb{i}] \vee \mathcal{A}_v^{i+1}[\mathbb{i}]) \wedge (\mathcal{A}_u^i[\mathbb{i}] \vee \neg\mathcal{A}_v^{i+1}[\mathbb{i}]) \qquad (16)$$

$$\bigwedge_{\mathbb{i}=1}^{\lceil \log_2(|A|+1) \rceil} \neg\mathcal{A}_u^i[\mathbb{i}] \vee \mathcal{M}_u^i \qquad (17)$$

Few additional constraints are needed to rule out extra values that can be represented by propositional vector but do not correspond to any agent if $|A|$ is not a power of 2.

Observe that all the constraints are now written as *clauses* (disjunctions of *literals*, where literal is a variable or its negation) or can be easily rewritten as clauses. Thus, a *conjunctive normal form* (*CNF*) [4] of the formula has been obtained. The resulting formula modeling existence of solution of given bCPF Σ with makespan bound η in the CNF form will be denoted as $F(\Sigma,\eta)$.

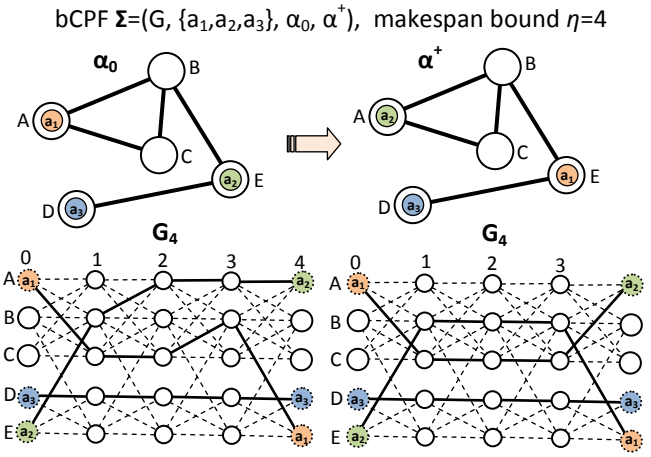


Fig. 4. ***Induced matchings*** *in time expansion graph connected into a sequence.* If a solution of a bCPF instance exists then there exists a sequence of consecutive matchings connecting the set of initial positions of agents with the set of their goals (*bottom left*) in the time expansion graph. The opposite however does not hold as sequence of matchings between sets of initial and goal positions disregards correspondence of the initial and goal position of individual agents (*bottom right*).

C. Properties of the Encoding

The constructed formula $F(\Sigma,\eta)$ has a *model* (a valuation of variables under which the formula is satisfied) if and only if Σ has a solution of makespan η. Moreover, a solution of Σ can be extracted from the model of $F(\Sigma,\eta)$ as it is summarized in the following proposition.

Proposition 2 (ENCODING CORRECTNESS). *The matching based encoding is correct. That is, there is a 1-to-1 mapping between solutions of bCPF* Σ *with bound* η *and models of* $F(\Sigma,\eta)$. ■

Sketch of proof. If there is a solution $\vec{s} = [\alpha_0, \alpha_1, \alpha_2, ..., \alpha_\eta]$ of given bCPF Σ then valuation of $\mathcal{A}_v^i$ can be directly constructed from arrangements $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_\eta$. Moreover, valid movements induce a matching which corresponds to valuation of $\mathcal{M}_v^i$ and $\mathcal{E}_{u,v}^i$. Altogether, constructed valuations constitute a model of $F(\Sigma, \eta)$.

The opposite direction is analogical. Assume that a model of $F(\Sigma, \eta)$ is given. Arrangements $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_\eta$ which constitute a solution are obtained from valuations of $\mathcal{A}_v^i$. Constraints and valuations of variables $\mathcal{M}_v^i$ and $\mathcal{E}_{u,v}^i$ ensure that $\mathcal{A}_v^i$ are non-zero in those vertices v and time steps i that are selected into matchings. Hence, validity conditions (ii) and (iii) are preserved. It is also ensured by constraints in $F(\Sigma, \eta)$ that valuations of $\mathcal{A}_v^i$ variables for consecutive time steps observe relocation of agents along edges selected into a matching. Thus, validity condition (i) is satisfied as well. ■

A very important advantage of matching encoding is its small size. Let us summarize the size of the encoding in the following proposition (the proof is omitted).

Proposition 3 (MATCHING ENCODING SIZE). *Let* $\Sigma = [G, A, \alpha_0, \alpha_+]$, *where* $G = (V, E)$ *with a bound* η *be an instance of bCPF. The induced part of the model requires* $\eta \cdot (|V| + |E|)$ ***propositional variables*** *and* $\eta \cdot (3|V| + 2|E|) + \eta \sum_{v \in V} (\deg_G(v) + 1) \cdot \deg_G(v)$ ***clauses***.

The mapping part of the model requires $\eta \cdot |V| \cdot \lceil \log_2(|A| + 1) \rceil$ ***propositional variables*** *and* $\eta \cdot (2|V| + |E|) \cdot \lceil \log_2(|A| + 1) \rceil$ ***clauses***. ■

The number of edges incident with a given vertex v is denoted as $\deg_G(v)$. Observe, that most of clauses in MATCHING encoding are either binary or ternary.

MATCHING encoding is smaller in terms of number of variables and clauses if compared with INVERSE and ALL-DIFFERENT encodings proposed in [21] and [22]. Even though the advantage in terms of size of the formula does not necessarily mean that the formula is easier to solve [5], certain correlation between solving time and the size of the formula exists. In order to determine whether this correlation actually exist in case of bCPF encodings, particularly in case of MATCHING encoding a thorough experimental evaluation must be done.

IV. SAT-BASED OPTIMAL CPF SOLVING

The suggested MATCHING encoding is intended for makespan optimal CPF solving. As it is possible to solve bCPF with given makespan bound η by translating it to SAT, an optimal makespan and corresponding solution can be obtained using multiple queries to a SAT solving procedure with encoded bCPF. Various strategies exist for getting the optimal makespan. The simplest one and very efficient one at the same time is to try sequentially makespan bounds $\eta = 1,2, ...$ until η equal to the optimal makespan is encountered. This strategy will be further referred as *sequential increasing*. The sequential increasing strategy is also used in domain independent planners such as SATPLAN [10], SASE [9] and others. Pseudo-code of the strategy is listed as Algorithm 1.

The focus here is on SAT encoding while querying strategies are out of scope of the paper; though let us mention that in depth study of querying strategies is given in [16]. There is a great potential in querying strategies as they can bring speedup of planning process in orders of magnitude, especially when combined with parallel processing.

Algorithm 1. SAT based optimal CPF solving.
input: a CPF instance Σ
output: a pair consisting of the optimal makespan and corresponding optimal solution

```
function Find-Optimal-Solution-Sequentially (Σ = (G, A, α0, α+)): pair
1:  η ← 1
2:  loop
3:      Ξ ← Encode-CPF-as-SAT (Σ, η)
4:      if Solve-SAT (Ξ) then
5:          s ← Extract-Solution-from-Valuation(Ξ)
6:          return (η, s)
7:      η ← η + 1
8:  return (∞, ∅)
```

Any complete SAT solver [8] may be used as the external module of the suggested optimal CPF solving algorithm. Notice however that a CPF solver following the framework of Algorithm 1 is *incomplete*. If the given CPF instance Σ has no solution then the algorithm runs infinitely. The treatment of incompleteness is easy. The solvability of Σ can be checked by some of sub-optimal polynomial time solving algorithms such as that suggested in [13] or by PUSH-AND-ROTATE [24] (which corrects previous algorithm [1]) before optimal SAT solving is started. The speed of the solving process is not compromised by solving the instance sub-optimally first since the runtime of solving encoded bCPFs by a SAT solver significantly dominates in the overall runtime.

V. EXPERIMENTAL EVALUATION

Properties of suggested MATCHING encoding were evaluated experimentally. Experimental findings regarding the size of the encoding and the speed of CPF solving when MATCHING encoding is used within the SAT based framework for makespan optimal CPF solving.

All the propositional encodings used in experimental evaluation (that is, MATHING, INVERSE [21], and ALL-DIFFERENT [22]) are further augmented with vertex reachability heuristic. The idea behind reachability heuristic is quite simple. If certain vertex v at time step i contains agent a while the distance from v to its goal is greater than the remaining number of time steps (that is, $> \eta - i$) then the instance is unsolvable from such a state. Hence, occurrence of an agent in vertices and time steps from that it is too far to the goal is forbidden. What is more important, constraints, in which particular agents are identified at forbidden vertices and time steps, can be omitted completely as they never contribute (this is however not the case of MATCHING encod-

ing since no particular agents are indentified there; in ALL-DIFFERENT encoding, the reachability heuristic reduces the size of resulting formula significantly).

`Glucose` version `3.0` [2] SAT solver has been used in the experimental evaluation. According to the 2013 SAT Competition [3] `Glucose` is one of few top SAT solvers in terms of performance in solving hard combinatorial problems. As CPF can be regarded as a combinatorial problem, this choice of SAT solver is justified.

All the source codes used to conduct experiments are posted on website to allow full reproducibility of presented results: http://ktiml.mff.cuni.cz/~surynek/research/ictai2014. Additional experimental results including data in raw form are on-line as well.

A. Comparison of Encoding Sizes

One of the objectives in designing the new CPF encoding was to reduce its size as much as possible. The size is considered in terms of the number of variables and clauses. Although it is reported that there is only a weak correlation between the formula size and the time necessary to test whether it is satisfiable [5], this is usually meant for domain independent case. In domain dependent case like CPF, we have a chance to solve the formula faster if there are fewer propositional variables and clauses in the encoding.

The size of suggested MATCHING encoding is compared with sizes of previous two encodings called INVERSE and ALL-DIFFERENT. Both previous encodings are much smaller than domain independent encodings used in planners SATPLAN and SASE as shown in [21], therefore, comparison with domain independent encodings is omitted.

Table 1. ***Encoding size*** *comparison* – 8×8. INVERSE, ALL-DIFFERENT, and MATCHING encodings – all with compiled distance heuristics [22] – are compared. bCPF instances are generated over the 4-connected grid of size 8×8 with 20% of cells occupied by obstacles. Makespan bound η is always 16. The number of variables and clauses is listed for different sizes of the of agents *A*. MATCHING encoding is smallest for larger number of agents in terms of number of variables and clauses.

Grid 8×8 |Agents|		INVERSE	ALL-DIFFERENT	MATCHING
1	#Variables	8 358.7	**1 489.3**	4 520.3
	#Clauses	31 327.9	**7 930.4**	25 881.1
4		10 019.5	7 834.5	**6 181.1**
		55 437.0	**34 781.9**	43 171.0
16		11 680.3	67 088.3	**7 841.9**
		91 344.5	216 745.4	**72 259.3**
32		12 510.7	230 753.0	**8 672.3**
		122 170.3	646 616.2	**99 675.5**

The comparison of encoding sizes is made on the standard benchmark used in CPF consisting of 4-connected grid graph with random initial and goal arrangements of agents [19]. Randomly placed obstacles occupy 20% of vertices. Comparison is made on grids of sizes 8×8, 12×12, and 16×16 with the size of the set of agents ranging from 1 to 32, 64, and 128 respectively. 10 random instances were generated for each size of the set of agents. A time limit of 1 minute has been used for generating the single formula (only the ALL-DIFFERENT encoding did not manage to generate formulae for large set of agents in the given time limit).

Selected results are shown in Table 1, 2, and 3. Average size calculated from 10 random instances is always presented. It can be observed that the ALL-DIFFERENT encoding is smallest for few agents in the graph, but it quickly blows up as the number of agents increases. Both INVERSE and MATCHING encodings grow relatively slowly with increasing number of agents while the MATCHING is consistently smaller than the INVERSE one.

Table 2. ***Encoding size*** *comparison* – 12×12. Makespan bound η is 24. The MATCHING encoding is again smallest for larger sets of agents.

Grid 12×12 |Agents|		INVERSE	ALL-DIFFERENT	MATCHING
1	#Variables	29 798.7	**4 973.9**	15 961.3
	#Clauses	116 302.8	**30 928.8**	94 603.2
4		35 381.1	22 190.0	**21 543.7**
		203 123.9	**122 571.6**	155 954.7
16		40 963.5	153 047.5	**27 126.1**
		330 613.4	632 067.5	**257 974.6**
32		43 754.7	475 135.0	**29 917.3**
		439 680.0	1 628 635.0	**354 306.4**
64		46 545.9	1 630 196.0	**32 708.5**
		620 942.7	4 713 520.0	**522 834.3**

As can be seen in section B, most of clauses in the MATCHING encoding are either **binary** or **ternary** (the ratio between the total number of literals appearing in the formula and the number of clauses is around 2.8), which suggests support for intensive *unit propagation*. INVERSE and ALL-DIFFERENT encodings has many higher-arity clauses when the number of agents is high. No such effect appears in MATCHING encoding with more agents.

Table 3. ***Encoding size*** *comparison* – 16×16. Makespan bound η is 32. The ALL-DIFFERENT encoding for 128 agents could not be generated in the given time limit of 1 minute.

Grid 16×16 |Agents|		INVERSE	ALL-DIFFERENT	MATCHING
1	#Variables	71 974.0	**11 413.6**	38 328.2
	#Clauses	286 764.5	**82 011.1**	230 572.1
4		85 094.0	**50 978.3**	51 448.2
		496 353.1	**336 001.7**	377 551.9
16		98 214.0	296 355.6	**64 568.2**
		803 130.0	1 521 163.0	**621 720.0**
32		104 774.0	847 829.1	**71 128.2**
		1 065 304.0	3 545 489.0	**852 589.4**
64		111 334.0	2 725 381.0	**77 688.2**
		1 498 740.0	9 320 047.0	**1 254 721.0**
128		134 035.5	N/A	**98 010.3**
		2 272 241.0		**1 996 918.0**

B. Comparison of Runtime[1]

The runtime comparison is focused on speed of makespan optimal CPF solving by SAT based framework. Again, speed of CPF solving when MATCHING encoding is used is compared with usage of previous two encodings – INVERSE and ALL-DIFFERENT. In order to show performance of SAT-

[1] All the runtime measurements were done on a machine with the 4-core CPU Xeon 2.0GHz and 12GB RAM under Linux kernel 3.5.0-48.

based optimal CPF in context, A* based solving algorithm OD+ID [20] has been implemented and included into comparison.

Table 4. *Average* ***optimal makespan*** *– grid 4×4.* As the number of agents increases the makespan increases as well. Agents need to interact with each other more intensively, which limits maneuverability.

Grid 4×4 |Agents|	1	2	3	4	5	6	7	8
Makespan	2.5	4.8	4.3	5.5	8.6	9.0	10.0	11.4

The experimental setup is the same as in the case of size evaluation, that is, a set of agents with random goals is randomly placed in a 4-connected grid of certain size which has 20% of its vertices occupied by obstacles. Grids of size 4×4, 8×8, and 12×12 were used. The time limit for the solver has been set to 1 minute. For each size of the grid a set of agents of the size ranging from 1 to a number of agents for which the problem was solvable in 1 minute (due to the time limit smaller instances than in size comparison have been used). Again, 10 instances are generated and solved for each size of the set of agents. Mixture of solvable and unsolvable instances has been used.

Table 5. *Average* ***optimal makespan*** *– grid 8×8.* The average distance to goals increased but the maneuverability in presented instances increased as well, hence the makespan is comparable to 4×4 grid.

Grid 8×8 |Agents|	1	2	4	6	8	10	12	16	20
Makespan	5.3	7.4	8.4	8.7	11.0	9.8	11.6	12.4	12.3

The first series of results regarding the optimal makespan is presented in Table 4, 5, and 6. The optimal makespan is correlated with the size of the grid since the distances between initial positions and goals are larger in larger grids. At the same time, the makespan is correlated with density of agents in the environment. If the density of agents is high the maneuverability of agents is limited which prolongs the makespan as a consequence.

Table 6. *Average* ***optimal makespan*** *– grid 12×12.* The average distance to goals has a major impact on the resulting makespan.

Grid 12×12 |Agents|	1	2	4	6	8	10	12	16	20
Makespan	11.3	13.0	14.0	15.5	17.5	18.0	33.8	18.7	21.0

Results regarding runtime are presented in Fig. 5, 6, and 7. It can be observed that A* base OD+ID performs as the worst. It is not able to solve the instance within the given time limit if the number of agents is higher and their relocation to goal requires non-trivial cooperation. The same weak performance of OD+ID is visible throughout all the three grids. However, as grids become larger the algorithm is able to solve instances with more agents since in larger graphs, there is better chance to have agents independent – a property OD+ID exploits.

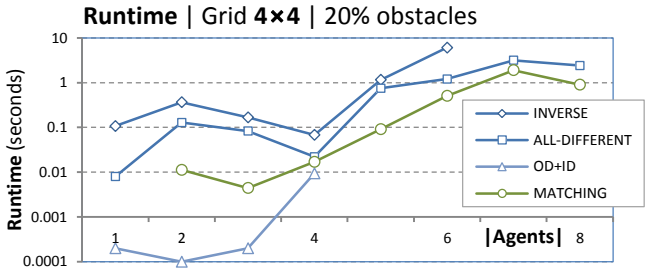


Fig. 5. ***Runtime*** *of SAT-based CPF solving – grid 4×4.* `Glucose 3.0` is used as an external solver. MATCHING encoding is up to the order of magnitude faster than other two encodings. A* based OD+ID is able to solve instances with few agents only.

On the other hand, OD+ID performs as the best when the number of agents is small. This is thanks to the design of the algorithm, which is able to plan paths for agents independently as shortest paths if they do not interact. Notice also that SAT based CPF solving uses quite complicated infrastructure, which has a certain overhead. For each number of time steps a new formula is generated which means to save it to a file. The SAT solver is then invoked and it needs to load the formula from the file. Nothing of this is needed in case of OD+ID as the algorithm runs everything in memory.

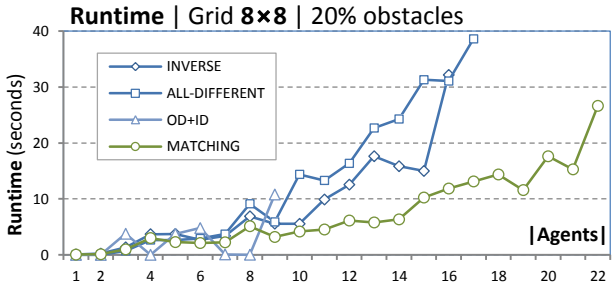


Fig. 6. ***Runtime*** *of SAT-based CPF solving – grid 8×8.* Dramatic difference between MATCHING encoding and the second best ALL-DIFFERENT encoding appears when occupancy of the graph with agents exceeds 15-16%.

The advantage of using SAT based solving becomes clear with larger sets of agents. The overhead of translating bCPF into propositional formula is outweighed by the ability of the SAT solver to solve very difficult combinatorial instances relatively quickly. Notice that dozens of techniques and heuristics implemented within the state-of-the-art solver are harnessed to the effort to solve a given bCPF instance. An important issue is memory management for example. Modern SAT employ elaborated techniques to keep memory requirements reasonable (such as deletion of learned clauses) while in case of A* based algorithms we face shortage of memory in say several minutes after starting the algorithm (all the visited states need to be stored into the *closed-list*).

When it comes to comparison of different encodings within SAT based solving, it is clear that INVERSE encoding performs as worst – it solved fewest instances and is slowest. Both ALL-DIFFERENT and MATCHING encodings perform much better while MATCHING encoding is the best in all the tests. Experiments suggest that MATCHING encoding is up to

twice as faster than the ALL-DIFFERENT encoding in larger graphs. The dominance of MATCHING encoding seems to start at 15-16% occupancy of the environment with agents (also we need to account 20% of static obstacles which gives 25-26% of total occupancy). The explanation is that ALL-DIFFERENT encoding grows quickly in terms of size with growing number of agents. The growth in case of MATCHING encoding is slow and it is accounted only to the distance heuristic, which need to handle more agents, and to initial and goal constraints (the rest of the formula is independent of the particular set of agents).

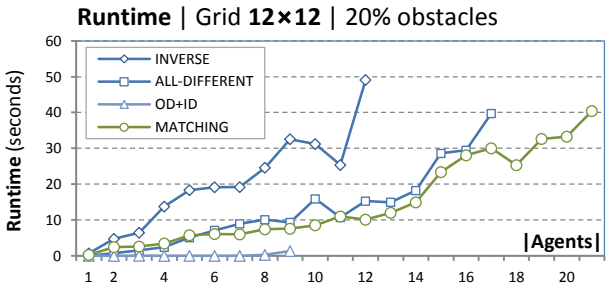


Fig. 7. ***Runtime*** *of SAT-based CPF solving – grid 12×12.* The point where MATCHING encoding becomes significantly fastest is shifted towards larger sets of agents (where timeout of 1 minute elapsed for other encodings).

VI. CONCLUSIONS AND DISCUSSION

A novel propositional encoding of cooperative path-finding problem (CPF) has been introduced. It is based on the concept of matching in a bipartite graph. Spatial (validity) constraints of CPF are modeled by connecting matchings in the time expansion graph into a sequence. Individual agents are considered in a separate part of the model, which allows keeping spatial constraints very simple and compact in the propositional representation. At the same time, the part of the model dealing with individual agents does not need to care about spatial constraints, which again brings significant simplification. Both sub-models are integrated by simple channeling constraints.

All these ideas let to the encoding – called MATCHING – that is small in the terms of number of variables and clauses. Consequently, the encoding can be solved faster, which improved the performance of the whole SAT-based solving process for CPF. MATCHING encoding let up to several times faster makespan optimal CPF solving than previous propositional encodings. The comparison with A* based methods shown that SAT based methods with any encoding dominate on CPF instances with relatively high density of agents.

There are several open questions for future. It would be interesting to see performance of CPF solving if it is modeled as a constraint satisfaction problem (CSP) or an integer-programming instance.

The suggested MATCHING encoding is still far from theoretically smallest possible encoding in which it is needed to represent locations all the agents at all the time steps. Consider that each vertex can be visited by an agent at most $\eta/2$-times where η is the makespan bound in the given bCPF.

REFERENCES

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

[2] Audemard, G., Simon, L. "The Glucose SAT Solver", http://labri.fr/perso/lsimon/glucose/, 2013, [accessed in May 2014].

[3] Balint, A., Belov, A., Heule, M., and Järvisalo, M. "SAT 2013 competition", http://satcompetition.org/, 2013, [accessed in May 2014].

[4] Biere, A., Heule, M., van Maaren, H., Walsh, T. "Handbook of Satisfiability", IOS Press, 2009.

[5] Bjork, M. "Successful SAT Encoding Techniques", Journal on Satisfiability, Boolean Modeling and Computation, Addendum, IOS Press, 2009.

[6] Dechter, R.: "Constraint Processing", Morgan Kaufmann Publishers, 2003.

[7] Dotú, I., del Val, A., Cebrián, M. "Channeling Constraints and Value Ordering in the QuasiGroup Completion Problem", Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann 2003, pp. 1372-1373.

[8] Eén, N., Sörensson, N. "An Extensible SAT-solver", Proceedings of Theory and Applications of Satisfiability Testing (SAT 2003), LNCS 2919, Springer, 2004, pp. 502-518.

[9] Huang, R., Chen, Y., Zhang, W. "A Novel Transition Based Encoding Scheme for Planning as Satisfiability", Proceedings of AAAI 2010, AAAI Press, 2010.

[10] Kautz, H., Selman, B. "Unifying SAT-based and Graph-based Planning", Proceedings of IJCAI 1999, Morgan Kaufmann, 1999, pp. 318-325.

[11] Luna, R., Berkis, K., E. "Push-and-Swap: Fast Cooperative Path-Finding with Completeness Guarantees", Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), IJCAI/AAAI Press, 2011, pp. 294-300.

[12] Korf, R. E., Taylor, L. A. "Finding Optimal Solutions to the 24-Puzzle", Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996), AAAI Press, 1996, pp. 1202-1207.

[13] 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), IEEE Press, 1984, pp. 241-250.

[14] Ratner, D., Warmuth, M. K. "Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable", Proceedings of AAAI 1986, Morgan Kaufmann, 1986, pp. 168-172.

[15] Régin, J-C. "A Filtering Algorithm for Constraints of Difference in CSPs", Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994), AAAI Press, 1994, pp. 362-367.

[16] Rintanen, J., Heljanko, K., and Niemelä, I. "Planning as satisfiability: parallel plans and algorithms for plan search", Artificial Intelligence, Volume 170 (12-13), Elsevier, 2006, pp. 1031–1080.

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

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

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

[20] Standley, T. S., Korf, R. E. "Complete Algorithms for Cooperative Pathfinding Problems", Proceedings of IJCAI 2011, IJCAI/AAAI Press, 2011, pp. 668-673.

[21] 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), LNCS 7458, Springer, 2012, pp. 564-576.

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

[23] Wang, K. C., Botea, A. "MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees", JAIR, Volume 42, AAAI Press, 2011, pp. 55-90.

[24] de Wilde, B., ter Mors, A., Witteveen, C. "Push and rotate: cooperative multi-agent path planning", Proceedings of International conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013), IFAAMAS, 2013, pp. 87-94.