A Simple Approach to Solving Cooperative Path-Finding as Propositional Satisfiability Works Well

Pavel Surynek
Charles University Prague, Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, 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). A novel very simple SAT encoding of CPF is proposed and compared with existing elaborate encodings. The conducted experimental evaluation shown that the simple design of the encoding allows solving it faster than existing encodings for CPF in cases with higher density of agents.

Keywords: cooperative path-finding (CPF), propositional satisfiability (SAT), SAT encodings, A*

1 Introduction and Context

The problem of *cooperative path-finding* (CPF) [12, 14] 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. The centralized planning mechanism finds a spatial-temporal path for each agent through which the agent can relocate to its goal. The 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. 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 [14].

Contemporary approaches to solving CPF include polynomial time sub-optimal algorithms [18] as well as methods that generate optimal solutions in certain sense [15, 16]. 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) [16, 17], methods based on

This work is supported by the Czech Science Foundation (contract no. GAP103/10/1287).

conflict resolution between paths for individual agents [13], and classical *$A*$ based methods* equipped with powerful heuristics [15]. 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.

This work tries to contribute to SAT-based methods. Particularly, it is inspired by [16] and [17] where quite complex and elaborate propositional encodings called INVERSE and ALL-DIFFERENT were proposed. The question here has been what would happen if a straightforward design of the encoding is adopted.

2 Cooperative Path Planning and Related Questions

An arbitrary **undirected graph** $G = (V, E)$ can be used to model the environment where agents are moving. Let A be a finite set of *agents*. Then, an arrangement of agents in vertices of graph G will be fully described by a *location* function $\alpha: A \longrightarrow V$; the interpretation is that an agent $a \in A$ is located in a vertex $\alpha(a)$. A generalized inverse of α denoted as $\alpha^{-1}: V \longrightarrow A \cup \{\bot\}$.

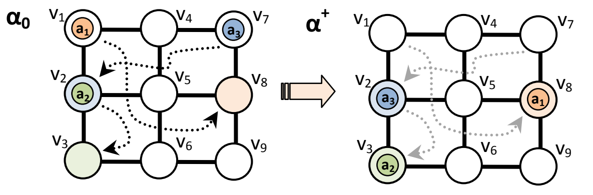
Definition 1 (COOPERATIVE PATH FINDING). An instance of *cooperative path-finding* problem is a quadruple $\Sigma = [G = (V, E), A, \alpha_0, \alpha^+]$ where location functions α_0 and α^+ define the initial and the goal arrangement of a set of agents A in G respectively. □

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 *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) $\forall a \in A \;\; \alpha_i(a) \neq \alpha_{i+1}(a) \Rightarrow \alpha_i^{-1}(\alpha_{i+1}(a)) = \bot$ (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)



CPF Σ=(G, {a_1,a_2,a_3}, α_0, α^+)

$\vec{s}$	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.

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 transformation of α_i for every $= 1,2, \ldots, m-1$. □

The task in CPF is to transform α_0 using above valid transitions to α_+. An illustration of CPF and its solution is depicted in Figure 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 [10].

3 A Simple SAT Encoding of the Bounded Variant of CPF

Let us describe a simple encoding called DIRECT of bCPF $\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$ with makespan bound η where $V = \{v_1, v_2, \ldots, v_n\}$ and $A = \{a_1, a_2, \ldots, a_\mu\}$ with $n, \mu \in \mathbb{N}$. Arrangements of agents over the graph at all the time steps from 1 to μ will be represented (that is, the graph will be expanded over time). The encoding will use a propositional variable for each vertex, agent, and a time step which will be assigned $TRUE$, if and only if the given agent appears in a given vertex at given time step.

Definition 3 (DIRECT ENCODING). A DIRECT encoding of a given bCPF $\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$ with makespan bound η consists of propositional variables $\mathcal{X}_{j,k}^i$ for every $i = 0,1, \ldots, \eta$, $j = 0,1, \ldots, n$, $k = 0,1, \ldots, \mu$. The interpretation is that $\mathcal{X}_{j,k}^i$ is assigned $TRUE$ if and only if a_k appears in v_j at time step i. The following constraints modeling validity conditions on consecutive arrangements are introduced:

(a) $\bigwedge_{j,l=1,j<l}^{n} \neg\mathcal{X}_{j,k}^i \vee \neg\mathcal{X}_{l,k}^i$ for every $i \in \{0,1, \ldots, \eta\}$, (4)
$\bigvee_{j=1}^{n} \mathcal{X}_{j,k}^i$ and $k \in \{1,2, \ldots, \mu\}$
(an agent is placed in exactly one vertex at each time step)

(b) $\bigwedge_{k,h=1,k<h}^{\mu} \neg\mathcal{X}_{j,k}^i \vee \neg\mathcal{X}_{j,h}^i$ for every $i \in \{0,1, \ldots, \eta\}$, (5)
and $j \in \{1,2, \ldots, n\}$
(at most one agent is placed in each vertex at each time step)

(c) $\mathcal{X}_{j,k}^i \Rightarrow \mathcal{X}_{j,k}^{i+1} \vee \bigvee_{l:\{v_j,v_l\}\in E} \mathcal{X}_{l,k}^{i+1}$ for every $i \in \{0,1, \ldots, \eta-1\}$, (6)
$\mathcal{X}_{j,k}^{i+1} \Rightarrow \mathcal{X}_{j,k}^i \vee \bigvee_{l:\{v_j,v_l\}\in E} \mathcal{X}_{l,k}^i$ $j \in \{1,2, \ldots, n\}$, and $k \in \{1,2, \ldots, \mu\}$
(an agent relocates to some of its neighbors or makes no move)

(d) $\mathcal{X}_{j,k}^i \wedge \mathcal{X}_{l,k}^{i+1} \Rightarrow \bigwedge_{h=1}^{\mu} \neg\mathcal{X}_{l,h}^i \wedge \bigwedge_{h=1}^{\mu} \neg\mathcal{X}_{l,h}^{i+1}$ (7)
for every $i \in \{0,1, \ldots, \eta-1\}$, $j, l \in \{1,2, \ldots, n\}$ such that $\{v_j, v_l\} \in E$ and $k \in \{1,2, \ldots, \mu\}$
(target vertex of a move must be vacant and the source vertex will be vacant after the move is performed). □

Observe that a *conjunctive normal form* (*CNF*) [2] of the formula has been obtained; it will be denoted as $F_{DIR}(\Sigma, \eta)$.

SAT-Based Optimal CPF Solving

The suggested DIRECT 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 solver 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 $\eta = 1,2,\ldots$ until η equal to the optimal makespan is encountered. The sequential increasing strategy is also used in domain independent planners such as SATPLAN [8], SASE [7] and others. Before a makespan optimal solution is searched, the solvability of the CPF instance should be checked by some of fast sub-optimal polynomial time solving algorithms such as PUSH-AND-ROTATE [18].

4 Experimental Evaluation

The proposed DIRECT encoding has been competitively evaluated with respect to other existing two propositional encodings of bCPF called INVERSE [16] and ALL-DIFFERENT [17]. Various static characteristics of encodings such as its size and runtime behavior were compared. The SAT-based solving has been compared with another state-of-the-art method developed around A* algorithm called OD+ID [15].

Table 1. ***Static characteristics*** *of encodings over 8×8 grid.* INVERSE, ALL-DIFFERENT, and DIRECT encodings are compared. bCPF instances are generated over the 4-connected grid of size 8×8 with 10% of cells occupied by obstacles. Makespan bound η 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. DIRECT encoding is biggest in terms of the length of formula but has smallest clauses in average and is most constrained out of all the encodings.

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

The experimental setup uses random CPF instances over 4-connected grids with randomly placed obstacles. This is a standard benchmark for evaluating CPF solving methods suggested in [14]. Initial locations and goals of agents were distributed randomly over the grid. Grids of sizes 6×6, 8×8, and 12×12 were used in experiments; 10% were occupied by obstacles. All CPF the instances were solvable. `Glucose` version `3.0` [1] SAT solver has been used in the experimental evaluation. 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/pricai2014.

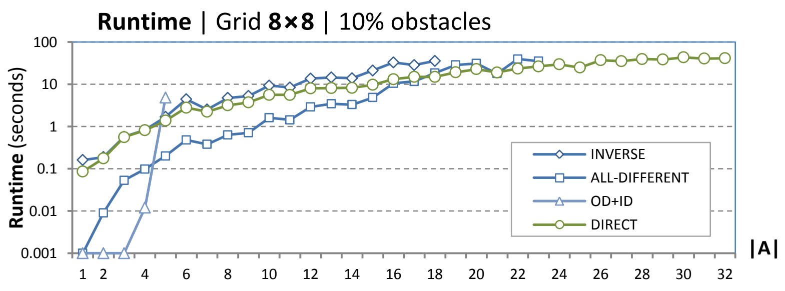
Static Evaluation of Encodings

There are several static characteristics of propositional formulae in CNF that are correlated with performance of their solving by most SAT solvers. The size of the formula in terms of the *number of variables* and the *number of clauses* determines the time needed to find a solution significantly.

Static characteristics of the DIRECT encoding are compared with other two propositional encodings over the 8×8 grid – INVERSE and ALL-DIFFERENT – in Table 1. Results for various numbers of agents are shown. The winner according to each characteristic is shown in bold (short clauses, high constrainedness, small formula are preferred). It can be observed that the smallest encoding in terms of the number of variables and clauses is the INVERSE one while the biggest one is the DIRECT encoding with ALL-DIFFERENT encoding standing in the middle. However in terms of the clause to variable ratio and the size of clauses, the DIRECT encoding seems to be the best as it has highest number of shortest clauses.

Runtime Evaluation of Encodings

The speed of SAT-based optimal CPF solving with the three discussed encodings has been evaluated. Again, 4-connected grids of various sizes were used in experiments. The *runtime*[1] needed for finding an optimal solution has been measured for the number of agents ranging from 1 to half of the number of vertices in the graph. The timeout of 1 minute has been used (since the increasing number of agents makes the CPF instance more difficult, the timeout has been reached before the limit of the number of agents in the largest grid). For each number of agents, 10 random instances of bCPF have been generated and solved.



Grid 8×8 / |A|	1	2	4	8	12	16	20	24	32
Makespan	5.6	5.3	8.0	9.7	10.8	11.0	11.5	11.5	10.4

Figure 2. ***Runtime*** *of SAT-based CPF solving – grid 8×8.* `Glucose 3.0` is used as an external solver in SAT-based solving. For each number of agents, 10 random instances were solved and the average runtime is reported. The DIRECT encoding can be solved the fastest for the higher occupancy with agents and it is the only encoding for which all the instances have been solved in a given timeout of 1 minute. The average optimal makespan for selected numbers of agents is shown in the table in the bottom. Note that OD+ID is fastest for sparsely populated graphs but its increases runtime quickly with higher number of agents.

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

Evaluation of SAT-based CPF solving would be incomplete if it is not compared with other state-of-the-art solving methods. Therefore A*-based method OD+ID [7] is included into competitive comparison.

Runtime results for the 8×8 grid are shown in Figure 2. Also average optimal makespans are shown. In case of grids of size 6×6 and 8×8, the SAT-based solving with the DIRECT encoding is the best option if the occupancy of the graph with agents is $> 30\%$. That is, the DIRECT encoding is a better option for more difficult instances on these two grids. The closest competitor to the DIRECT encoding is the ALL-DIFFERENT encoding which is a better option for less occupied graphs. In very sparsely occupied graphs, OD+ID method is the best as lot of independence among agents can be found. However, OD+ID degrades dramatically if there is higher concentration of agents in the graph since agents become more interdependent and independence heuristics no longer work.

The INVERSE encoding was always the worst option out of all the tested methods. We consider that the reason for its weak performance is that relatively long clauses appear in it. On the other hand, short clauses of the DIRECT encoding and their abundance promoting unit propagation are the main reasons for the good performance of this encoding. We observed that solving of formulae of the DIRECT encoding by the SAT solver is relatively fast while large portion of the time is consumed by generating the formula (the formula is generated into file, which is subsequently read by the SAT solver). Hence, there is still room to increase the speed of SAT-based solving if the solving process is better engineered.

6 Discussion, Conclusions, and Future Works

A new propositional encoding of the makespan bounded cooperative path-finding problem (bCPF) has been proposed. The idea of the work was to design very simple encoding with no elaborate technique behind and to check how it stands with respect to existing relatively elaborate encodings for the problem. The new encoding has been called DIRECT as it encodes the bCPF problem in the most straightforward way we were able to imagine.

The DIRECT encoding has been used within the SAT-based framework for solving CPF (unbounded version) optimally. The comparison with existing two encodings INVERSE [16] and ALL-DIFFERENT [17] as well as with A* search based method OD+ID [15] on random CPF instances over 4-connected grids has been done and showed surprising results. The DIRECT encoding despite its relatively naive design performed better than the ALL-DIFFERENT encoding on instances with occupancy by agents $> 30\%$ and almost always better than the INVERSE encoding.

Generally, the SAT-based approach turned out to be better whatever encoding has been used than the A* based OD+ID whenever occupancy with agents has been higher than trivial. This can be explained by the fact that OD+ID's heuristic cannot detect independence among agents. Note also that this method can be regarded as all-in-one while in the SAT-based approach the SAT solver itself is external. It is unrealistic to implement equivalent number of propagation, learning, and heuristic techniques in the all-in-one solution as they are in SAT solvers, which we can access through encoding the problem in this formalism in a relatively simple way.

References

1. Audemard, G., Simon, L. *The Glucose SAT Solver.* http://labri.fr/perso/lsimon/glucose/, 2013, [accessed in June 2014].
2. Biere, A., Heule, M., van Maaren, H., Walsh, T. *Handbook of Satisfiability.* IOS Press, 2009.
3. Bjork, M. *Successful SAT Encoding Techniques.* Journal on Satisfiability, Boolean Modeling and Computation, Addendum, IOS Press, 2009.
4. Dechter, R. *Constraint Processing.* Morgan Kaufmann Publishers, 2003.
5. Eén, N., Sörensson, N. *An Extensible SAT-solver.* Proceedings of Theory and Applications of Satisfiability Testing (SAT 2003), pp. 502-518, LNCS 2919, Springer, 2004.
6. Gent, I. P., Walsh, T. *The SAT Phase Transition.* Proceedings of the 11th European Conference on Artificial Intelligence (ECAI 1994), pp. 105–109, John Wiley and Sons, 1994.
7. Huang, R., Chen, Y., Zhang, W. *A Novel Transition Based Encoding Scheme for Planning as Satisfiability.* Proceedings AAAI 2010, AAAI Press, 2010.
8. Kautz, H., Selman, B. *Unifying SAT-based and Graph-based Planning.* Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 318-325, Morgan Kaufmann, 1999.
9. Korf, R. E., Taylor, L. A. *Finding Optimal Solutions to the 24-Puzzle.* Proceedings of the 13th National Conference on Artificial In-telligence (AAAI 1996), pp. 1202-1207, AAAI Press, 1996.
10. Ratner, D., Warmuth, M. K. *Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable.* Proceedings of AAAI 1986, pp. 168-172, Morgan Kaufmann, 1986.
11. Régin, J-C. *A Filtering Algorithm for Constraints of Difference in CSPs.* Proceedings of the 12th National Conference on Artificial In-telligence (AAAI 1994), pp. 362-367, AAAI Press, 1994.
12. Ryan, M. R. K. *Exploiting Subgraph Structure in Multi-Robot Path Planning.* Journal of Artificial Intelligence Research (JAIR), Volume 31, pp. 497-542, AAA Press, 2008.
13. 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.
14. Silver, D. *Cooperative Pathfinding.* Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005), pp. 117-122, AAAI Press, 2005.
15. Standley, T. S., Korf, R. E. *Complete Algorithms for Cooperative Pathfinding Problems.* Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), 668-673, IJCAI/AAAI Press, 2011.
16. 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, pp. 564-576, Springer, 2012.
17. 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 Press, 2012.
18. 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), pp. 87-94, IFAAMAS, 2013.