Using Incomplete Satisfiability Modulo Theories to Determine Robotic Tasks*

Andreas Witsch, Hendrik Skubch, Stefan Niemczyk, and Kurt Geihs[1]

***Abstract*— Many robotic task specifications can be naturally expressed by boolean combinations of arbitrary constraints. This allows a separation of problem description and solution strategy. In this paper, we present a novel approach to solve non-linear constraint systems based on Satisfiability Modulo Theories. While most SMT-based techniques emphasize completeness, we intentionally use an incomplete local search strategy. Despite this incompleteness, the presented solution is able to deal with many real world problems like task allocation or robot positioning. We show that our approach is able to exploit the logical structure to solve highly complex tasks almost in real-time.**

I. INTRODUCTION

One of the main problems tied to the employment of Multi-Robot Systems (MRS) is the task allocation problem [1], which an adaptive and extensible MRS has to solve on-the-fly. While a solution to the task allocation problem determines which robot takes on which task of a given problem, actually completing an individual task often requires solving sub-problems, such as calculating a position or a trajectory. We refer to these sub-problems as task determination problems. In complex scenarios, the individual sub-problems are no longer independent: They reference each other. For example, when multiple robots lift an object, each individual robot's position depends on the others. We therefore consider a unified approach, in which the task allocation problem is solved *with respect to* all numerical sub-problems involved. The result constitutes not only a solution to the task allocation problem, but also to the sub-problems. In other words, it contains an answer substitution for all involved variables. In many cases these constraint satisfaction problems (CSP) have multiple solutions. Thus, a cost- or utility function can be incorporated to evaluate different solutions against each other in order to find an optimal variable assignment. The resulting constraint optimization problem (COP) describes the agents' goal and therefore separates the algorithm to reach it. Such a separation can simplify the software complexity in various robotic scenarios but requires a general solver capable of dealing with non-linear expressions. These are usually caused by the spatial nature of many robot task descriptions.

The problem class we consider here incorporates transcendental functions, e.g., trigonometric and exponential functions, rendering the resulting problems undecidable in general [2]. However, these kinds of functions occur in many robotic scenarios involving, for instance, inverse kinematics or sensor models based on Gaussian probability distributions.

As major contribution of this paper we investigated a new, incomplete approach geared towards efficiency. Thereby we are exploiting the given logical problem structure by state of the art methods borrowed from the Boolean satisfiability (SAT) community. Furthermore, we will show that despite the incomplete nature of our algorithm, it is able to solve a wide variety of complex problems. Thus, our approach is applicable in highly dynamic domains such as robotic soccer. As evaluated later, complete state of the art approaches can either not deal with the required problem class or are not able to keep the runtime constraints.

This paper is structured as follows. After stating a formal definition of the targeted problem class in Section II, related approaches are discussed in Section III. In Section IV, we describe our approach based on local search. Afterwards, we evaluate its performance in six different problem formulations in Section V. Finally in Section VI, we conclude and sketch future research topics.

II. PROBLEM DEFINITION

The problems we tackle are continuous non-linear constraint satisfaction problems (CNLCSP) [3]. These have the following components:

- A set of n real-valued variables X,
- A propositional formula ϕ with the variables P, where each $p_i \in P$ identifies a constraint $c_i = f_i(\vec{x}) \circ_i g_i(\vec{y})$, where $\circ_i \in \{<, >, \leq, \geq, =, \neq\}$, $\vec{x} = x_1, \ldots, x_m$, $\vec{y} = y_1, \ldots, y_l$, all $x_j, y_j \in X$, and all f and g are arbitrary functions $\mathbb{R}^k \mapsto \mathbb{R}$.

An interpretation of a CNLCSP is a valuation function $v\colon X \mapsto \mathbb{R}$, which is extended to $P \mapsto \{\top, \bot\}$ by

$$v(p_i) = \begin{cases} \top & \text{if } v(f_i(\vec{x})) \circ_i v(g_i(\vec{y})) \\ \bot & \text{otherwise} \end{cases}$$

The interpretation v is a *solution* for formula ϕ iff ϕ', constructed by replacing all variables p_i in ϕ with their interpretation $v(p_i)$, evaluates to $\top$ under classical propositional interpretation.

As this problem class is in general undecidable [2], solvers either do not terminate or stop after a given termination criteria. The underlying Boolean SAT problem is already NP-complete [4]. However, recent research within the Satisfiability Modulo Theory (SMT) community allowed DPLL(T) algorithms [5] to tackle this problem class.

*The project IMPERA is funded by the German Space Agency (DLR, Grant number: 50RA1112) with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

[1]Faculty of Electrical Engineering and Computer Science, University of Kassel, 34121 Kassel, Germany {awi,hsk,sni,kge}@vs.uni-kassel.de

III. RELATED WORK

In [3], we presented a possible solution to CNLCSPs based on local search. Although this approach is incomplete, its high sampling rate allows us to solve a wide variety of common practical problems. This paper focuses on an extension of local searches by satisfiability modulo theories. While the SMT solver ABSolver by Bauer et al. [6] also employs incomplete local searches for solving non-linear continuous problems, it is not specifically designed to deal with undecideable problem classes. In contrast, incompleteness is considered explicitly in our solver.

Recent solvers are able to solve various non-linear problems based on Gröbner basis like z3 [7] from Microsoft Research. However, Gröbner basis systems only allow us to decide polynomial arithmetic, which render them unable to deal with more general geometric problems which can involve transcendental functions. In contrast to our approach, z3 has different built-in theory solvers that allow to deal with linear, Boolean, or integer problems efficiently.

A different approach to solving non-linear problems is based on interval propagation. These approaches compute a floating-point box, which contains possible solutions to the problem. ISat [8] is a prominent SMT solver based on this technique. The focus of iSat is to solve problems with a complex logical structure and is thus far not capable of processing rationals or arbitrary powers. Our approach makes use of interval propagation to shrink the search space to an upper bound, thus decreasing the impact of the incomplete nature of the used local search technique.

Many SMT solvers rely on on the DPLL algorithm [9] to solve the underlying propositional SAT problem. Based on this method and efficient clause learning heuristics, the minisat solver [10] achieved high success at various SAT competitions. Due to its effectiveness and its clear implementation, it served as a basis for our implementation.

The specification logic TAL (temporal action logic) [11], [12] follows ideas and motivations of using constraints in complex task allocation problems similar to ours. However, TAL focuses on the temporal aspects of the problem, whereas we emphasize spatial constraints and quick, yet incomplete, solving strategies.

IV. APPROACH

In this section we present the Carpe Noctem Satisfiability Modulo Theory (CNSMT) solver. The major contribution is the incorporation of a local search based theory solver with an efficient SAT solver, which allows to solve highly non-linear problems as found in robotic domains.

A. SAT Solver

In order to apply a DPLL algorithm, the first step is a transformation of the propositional formula into conjunctive normal form. Afterwards, a slightly modified DPLL scheme as shown in Algorithm 1 is applied.

In the first step of the main loop, the current propositional assignment is checked for conflicts (Line 2). The propagation method is based on the optimized Boolean constrained

```
1  while true do
2      while propagate() do
3          if Decisionlevel = 0 or !resolveConflict() then
4              deleteUnsafeConclusions()
5          end
6      end
7      if !IntervalPropagate(decisions) or
       !LocalSearch(decisions, intervals, lastSolution) then
8          Continue
9      end
10     if satisfied(decisions) then
11         return true
12     end
13     next := Decide()
14     ReduceClauseDB()
15 end
```

Algorithm 1: CNSMT Main-Loop

propagation proposed by [13]. Therefore, two literals of each clause are observed. If one of these literals becomes false, the observed literal is changed to a new unobserved literal. If the clause does not have unobserved literals, the second literal is assigned a value of true. This procedure assigns all implications of new variable assignments. Due to these implications, clauses might become unsatisfiable under the given assignment. We call these clauses *conflict clauses*.

Detected conflict clauses are resolved by a backtracking search within the *resolveConflict* operation. Thereby, new clauses are learned as proposed by the GRASP-scheme [14]. The basic idea is to add a new clause containing all decisions since the last unique implication point of the current conflict. Afterwards, all literals with reasons that are already included in the new clause are removed to simplify the clause, see [10]. The learned clauses speed up the search process as they help to avoid similar conflicts for future decisions.

If the CNLCSP is unsatisfiable due to its propositional structure, a learned clause will lead to a conflict at the top decision level. This is the case if a learned unit clause conflicts with another and renders conflict resolution impossible. As the theory solver is incomplete, unsatisfiability can only be proven in case of unit propagation based on interval propagation. Thus, we delete all learned clauses that result from the theory solver or resolved conflicts (see Line 4) instead of inferring unsatisfiability. In order to avoid infinite solution search in practice, the main loop should be canceled after a termination criteria is met, e.g., a given timeout or number of evaluations.

In contrast to standard SAT problems, assignments might not only conflict at propositional level, but also in the non-linear theory. We combine two theory solvers to enhance the solution detection, as shown in Line 7. In case of a conflict, the inverted current assignment is added as a new clause and the outer loop is restarted to resolve the resulting conflict. Interval propagation is a complete approach. The learned conflict clauses are stored separately because there is no need to delete them at any time. The second theory is a local

search technique which is initialized by the last successful solution respectivly by the computed solution interval (see Section IV-C).

The next step in Line 10 checks whether the current partial assignment is already a solution for the propositional problem. This is the case iff all problem clauses have at least one literal which can be evaluated to true given the current propositional assignment. Consequently, the last computed assignment of X solves the CNLCSP.

The final two steps are assignment decision and clause database reduction. As a decision heuristic we count the number of clauses satisfied by a given variable. This strategy leaves potential for improvement, as the evaluation is computationally expensive. However, in empirical experiments it performed better than other strategies like a *Variable State Independent Decaying Sum* (VSIDS) decisions heuristic [13]. The learned clause database increases the computation time needed to detect conflicts; therefore, we count the activity of learned clauses and eliminate rarely used clauses similar to [10].

B. Interval-Propagation

Interval propagation manages an interval for each real-valued term occurring in the problem. Using downward and upward refinement operators, these intervals are contracted until either no further refinement steps are possible or until an interval collapses, in which case the corresponding sub-problem has no solution.

As opposed to solvers such as iSAT [8], which completely rely on interval propagation (IP) in order to solve sub-problems in theory level, IP only augments our solution scheme. Instead of splitting intervals repeatedly, IP only provides initial intervals for the local search and rules out solution candidates which can be proven to be infeasible. Thus, IP is used in two steps of the solver. First, during the initialisation phase, IP is done for each distinct literal occurring in the problem. This step can discover some unit clauses early and provide pre-propagated intervals for the search phase. Second, whenever a conjunctive sub-problem is identified by the SAT solver on the path to a solution, IP is performed in order to either provide boundaries for the local search or reject the conjunction in case of infeasibility.

For the purpose of this paper, we employ a simple IP algorithm working on a tree-shaped representation of the input problems. More sophisticated techniques (c.f. [8], [15]) can potentially lead to a better overall performance.

C. Local Search

In [3], we presented a solver for the considered problem class based on local search. We could show that it outperformed state-of-the-art solvers in a wide variety of problems relevant to robotic domains. The solver simply transforms the CNLCSP into an error function and performs gradient ascent on the resulting function. Since the SAT solver queries only for solution of conjunctive problems, this transformation can be simplified. Each constraint is transformed by the following rules:

Definition 4.1: Formula transformation

$$\begin{aligned} T(a < b) &= <^*(a, b) \\ T(a > b) &= <^*(b, a) \\ T(\neg(a < b)) &= \leq^*(b, a) \\ T(\neg(a > b)) &= \leq^*(a, b) \\ T(\neg(a \leq b)) &= <^*(b, a) \\ T(\neg(a \geq b)) &= <^*(a, b) \end{aligned}$$

Definition 4.2: Atomic Constraint Function

$$<^*(a, b) = \begin{cases} 1 & \text{if } a < b \\ b - a & \text{otherwise} \end{cases}$$

$$\frac{\delta}{\delta x_i} <^*(a, b) = \begin{cases} 0 & \text{if } a < b \\ \frac{\delta}{\delta x_i}(b - a) & \text{otherwise} \end{cases}$$

Due to the deflation of the problem to a pure conjunctive problem, we need only a transfer function for the and-operator. For local searches, the $\min$-operator known from the Gödel T-Norm in Fuzzy Logic is an appropriate choice. However, as shown in [3], a sum-based approach outperforms the T-Norm operator:

$$\Sigma_\wedge(a, b) = \begin{cases} 1 & \text{if } a = 1 \wedge b = 1 \\ \min(0, a) + \min(0, b) & \text{otherwise} \end{cases}$$

$$\frac{\delta}{\delta x_i}\Sigma_\wedge(a, b) = \frac{\delta}{\delta x_i}(a + b)$$

$$T(p \wedge q) = \Sigma_\wedge(T(p), T(q))$$

As a local search method, we used gradient ascent enhanced by the resilient backpropagation algorithm (Rprop) [16]. The function evaluation and gradient direction is computed by an extended version of Alex Shtof's auto differentiation library for the .NET framework [17]. Note that the pure conjunctive form can potentially reduce many non-convex problems to convex ones, which benefits local search techniques. In order to reduce the impact of the starting point, we restart after 60 Rprop steps at a randomly sampled point within the current interval area.

D. Constraint Optimization

In many robotic examples, we are not only interested in a valid solution for the CNLCSP, but the best solution with respect to a given utility function. For example, in a task allocation problem, we might be interested in the assignment with the least power requirement.

For the optimization case we exploit the fact that the transformation rules of Definition 4.2 lead to a function with an upper bound of 1. Thus, we can apply the gradient ascent for an arbitrary utility function if its value is greater than 1 and the CNLCSP is satisfied.

In order to find the global optimum, all propositional solutions have to be analyzed. This can simply be achieved by adding a new clause for each analyzed solution, containing the negated current assignment of P until no new solution can be found. Consequently, the optimization algorithm

terminates when the SAT-solver determines the problem to be unsatisfiable due to the additional clauses.

V. EVALUATION

To show the performance of the presented solution we evaluate different problem classes with the following solvers:

- iSat – recent developer version of iSat [8],
- z3 – Microsoft's z3 solver [7],
- GSolver – our Rprop($\Sigma_\wedge$, max) solver presented in [3], which is a stand alone version of the descriped T-solver,
- CNSMT – the approach presented in this paper,
- CNSMT-noIP – the presented approach without interval propagation.

First, we present the 3-SAT sine problem, which is the target problem class of our approach as it combines propositional structure with highly non-linear arithmetic. Afterwards, we show a multi-agent task determination and allocation problem, which is less artificial and complex with respect to the required theories. Next, we use the problem of building ad-hoc communication chains to a given target using mobile robots. Such a problem might occur in search and rescue domains. Finally, we will combine this problem with a utility function to demonstrate the optimization case for multiple solutions. All experiments were performed single threaded on an Intel i7 930 CPU (2.8 GHz) using Linux 3.2.0-35 with Mono 2.10.8.1 and are averaged over 1000 trials. Note that the time measurement of iSat only has a precision of 10 ms.

A. 3-SAT Sine Problem

In a first experiment, we show the efficiency of our approach for solving highly non-linear problems given a complex underlying boolean structure. This benchmark was originally introduced in [18], based on [19]. Thus, we assume $n = 25$ variables ranging over the reals and $l = 50$ inequalities $p \in P$ of the form:

$$k\Sigma_{i=1}^{3}\Pi_{j=1}^{3} a_{ij} \sin(2\pi x_{ij} + b_{ij}) < \theta \tag{1}$$

with $k = \frac{1}{\Sigma_{i=1}^{3}\Pi_{j=1}^{3} a_{ij}}$. All a_{ij} are uniformly distributed random values in $[-1, 1]$, all b_{ij} uniformly distributed random values in $[-2\pi, 2\pi]$, and all x_{ij} randomly chosen variables in X. The value of θ is a threshold such that the feasible region of the constraint is approximately half the size of the whole domain measured by random sampling. This resembles the ratio by which the solution space in pure SAT problems is divided by a single propositional variable. The 3-SAT formula ϕ, consisting of m clauses, is constructed by randomly drawing from P.

Satisfiability of the CSP is guaranteed in the following way: Let $\vec{s}$ be a random point in $\mathbb{R}^d$, acting as valuation function v. Let P' be the set of propositional variables evaluated to $\top$ by the extension of v. Then, for every clause in ϕ, pick a random literal and set its sign to positive (negative) if its propositional variable is in P' (is not in P').

As shown in Figure 1, our approach outperforms GSolver and iSat for a constraint ratio (m/n) over 2. These results

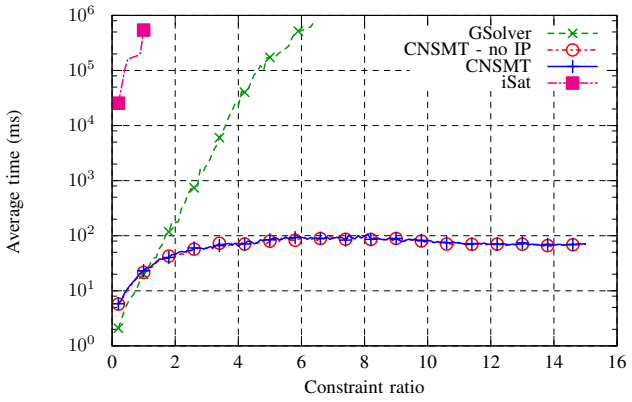

Fig. 1. 3-SAT Sine Test

follow the presumption we stated in [18]. Furthermore, we can also observe a phase transition at a constraint ratio of roughly 7.8, similar to the phase transition of SAT solvers for random 3-SAT problems [20]. Note that z3 cannot deal with trigonometric functions and is therefore excluded from this experiment. Even though this case is artificial it shows the performance given a beneficial logical structure, as the runtime stays constant for a constraint ratio of about 2.

B. Task-Determination and -Allocation

In order to present a more practical problem, we evaluated our approach for a task determination and allocation problem for a team of robots. The goal is to determine n positions p_i and assign them to n robot target positions t_i. Expressed as:

$$\bigwedge_{i=1}^{n}\bigvee_{k=1}^{n}(t_i = p_k).$$

where each position is a two dimensional point randomly distributed in a $10 \times 10\,\mathrm{m}$ area. Equality constraints are expressed as: $(t_{i_x} - p_{j_x})^2 + (t_{i_y} - p_{j_y})^2 < 0.05^2$.

Increasing the difficulty of the problem, we required an exclusive position for each robot. We achieved mutual exclusion by three different formulations, resulting in three semantically equivalent problems:

$$\bigwedge_{i=1}^{n}\bigvee_{k=1}^{n}(t_k = p_i) \tag{2}$$

$$\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{n}\bigwedge_{k=1}^{i-1}(t_i \neq p_j \vee t_k \neq p_j) \tag{3}$$

$$\bigwedge_{i=1}^{n}\bigwedge_{k=1}^{i-1}(t_i \neq t_k) \tag{4}$$

Constraint 2 requires each position to be assigned to a robot. Since the number of robots and positions are equal, the only valid assignment must be an exclusive position for each robot. Furthermore, the number of clauses is n with $n - 1$ disjunctions each and has the least memory requirement. Figure 2 shows that the CNSMT solver performs poorly compared to other solutions. This results from the fact that the formulation does not allow to draw multiple conclusions

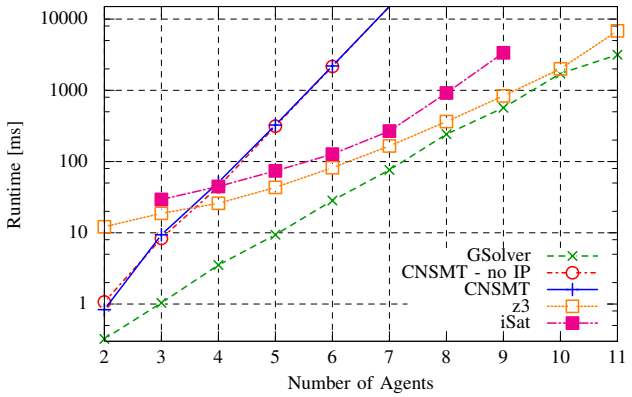

Fig. 2. Required Runtime to Solve Constraint 2

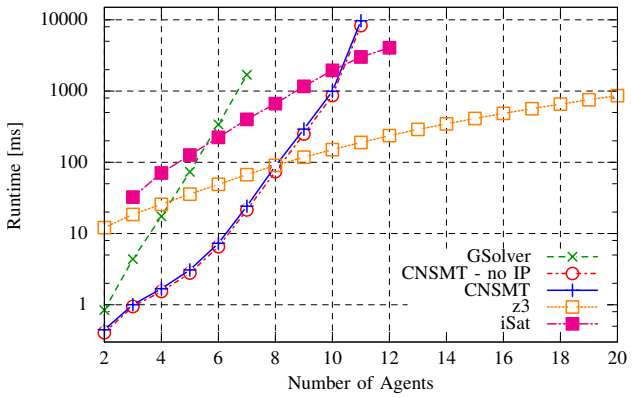

Fig. 3. Required Runtime to Solve Constraint 3

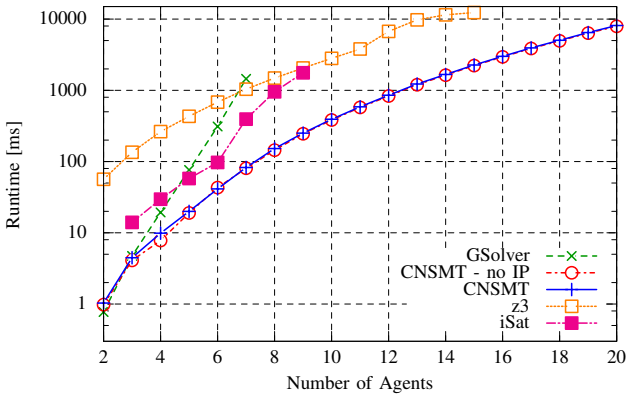

Fig. 4. Required Runtime to Solve Constraint 4

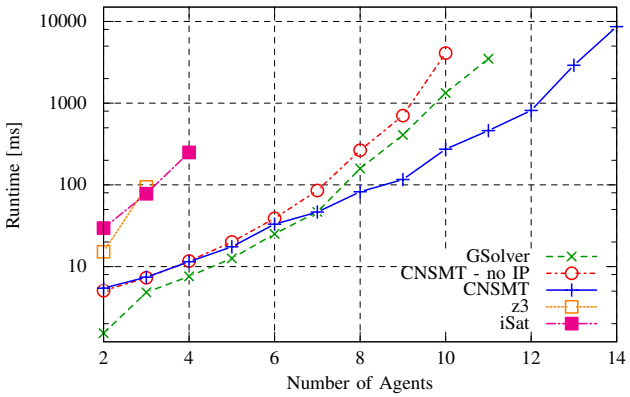

Fig. 5. Required Runtime to Compute Communication Chains

on the propositional level. Instead, multiple T-solver queries are required until a single propositional conflict can be discovered. In contrast, GSolver directly solves the complete expression, which results in the best overall performance.

In contrast to Constraint 2, Constraint 3 requires for each target position and each pair of robots that one of them is not assigned to that position. This yields $n^3/2$ clauses with only two literals, a structure very suitable for SAT solving, but with high memory requirements.Therefore, the SAT-based solvers outperform the GSolver, as shown in Figure 3. Furthermore, this experiment shows the small overhead of CNSMT compared to z3 or iSat, which is manifested by the small runtime for $n < 9$. On the other hand, z3 and iSAT scale better than CNSMT and start to outperform it at 8 and 11 agents, respectively. However, we deem a good performance for small problems to be essential in order to enable reactive behavior of the multi-robot system.

Constraint 4 requires all target positions to be mutually different. In contrast to Constraint 3, this requires only $n^2/2$ clauses. Figure 4 shows how CNSMT outperforms all other solutions. The fast responses of the T-solver and the high number of theory queries allow to reject conflicting assignments and learn expressive clauses quickly. Even though iSat and z3 use a similar technique, their non-linear arithmetics solvers perform considerably worse. Note that GSolver, CNSMT, and CNSMT-noIP solved all presented problem instances. That shows the robustness of the stated approaches despite their incompleteness.

C. Communication Chains

In order to show the power of interval propagation, we utilize an experiment with increased dependency between the agents. Here, the goal of the n involved agents, each equipped with wireless communication modules and routing capabilities, is to extend the communication range of a stationary base station at a position b to a target position g. The CSP describes n 2-dimensional points P_n with $|p_i - p_{i+1}|_2 < 100\,\mathrm{m}\ \forall i < n$. The last agent is placed at the target position $p_n = g$. To keep contact with the base station one robot has to stay in communication range $\bigvee_{i=1}^{n} |p_i - b|_2 < 100\,\mathrm{m}$. Base station and target position are separated by two impassable chains of mountains represented by rectangular obstacles. The valley in between has a length of $120\,\mathrm{m}$ to force a robot placement in this area. Valley position and relative angle between b and g are chosen randomly for each trial and the distance is set to $90 \cdot n\,\mathrm{m}$.

As shown in Figure 5 CNSMT again outperforms all other approaches as it scales best with the number of agents. In particular, interval propagation increases the performance for more than 3 agents. Both complete solver z3 and iSat are not able to solve the problem for more than 3 and 4 agents respectively within the given time constraints.

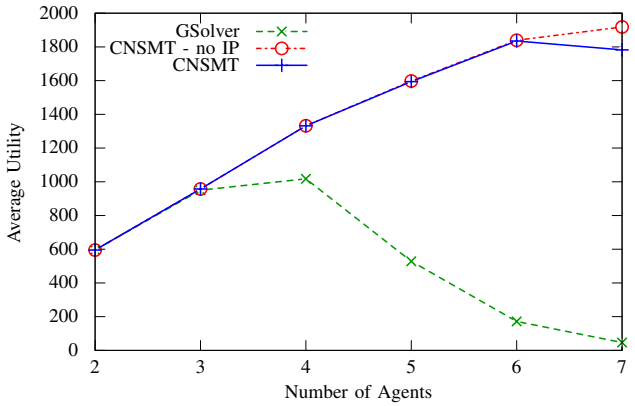

Fig. 6. Average Utility Value when Solving Constraint Optimization Problem 5

D. Constraint Optimization

Section V-B shows another typical problem in multi-agent scenarios: In each situation, multiple possible solutions exist. In most cases we want to distinguish between these using a utility or cost function and expressing an expected performance measure for the corresponding assignment. We formalized a constrained optimization problem by incorporating the following utility into the test case given by Constraint 3:

$$U(\vec{t}, \vec{x}) = nc_{\text{maxDist}} - \text{dist}(\vec{t}, \vec{x}), \tag{5}$$

where $\vec{x}$ represents the current physical positions of the n agents and c_{maxDist} the maximum possible distance in the target area to force a lower bound greater 0. Maximizing this function leads to an assignment, which minimizes the agents' travel distances and therefore the total energy costs, assuming homogeneous robots and terrain. Figure 6 shows the average utility with respect to the number of agents when assuming $30\,\text{ms}$ available computation time. The elimination of analyzed propositional solutions allows a more systematic local search and therefore leads to a broader exploration. Note that the optimization process of each solution aborts after a single exploration. Thus, optimization problems with more complex solution spaces might decrease the performance or require a sophisticated heuristic for a lower bound.

VI. CONCLUSIONS

In this paper, we presented an incomplete solver for non-linear continuous constraint satisfaction problems. We evaluated this technique and similar state-of-the-art solvers on problems originating in multi-robot system domains. We showed that highly complex CNLCSPs can be solved in an efficient manner by using an incomplete T-solver. This property makes our approach a practical tool for separating the problem description and solution strategy when designing and implementing robotic systems.

In future research additional heuristics will be investigated. As indicated in Section V-B the runtime can be decreased for some problems because there is no need to call the T-solver for each variable assignment. Furthermore, we used a very simple variable assignment strategy which might lead to long backtracking paths. Additional new heuristics are needed to decide when to delete a learned clause. The incompleteness of the T-solver plays an important role in this decision.

Finally, the task allocation experiment showed the importance of the problem formulation with respect to the solver. Analyzing this dependency could be very beneficial for future strategies that reformulate a given problem based on the available solver or choose a solver based on the formulation.

Further research can tackle distributed constraint solving along the lines of [21] and [18]. This could lead to a valuable runtime improvement for multi-agent scenarios.

REFERENCES

[1] B. P. Gerkey, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, 2004.

[2] D. Richardson, "Some Unsolvable Problems Involving Elementary Functions of a Real Variable," *Journal of Symbolic Logic*, 1968.

[3] H. Skubch, "Solving non-linear arithmetic constraints in soft realtime environments," in *27th Symposium On Applied Computing*. ACM, 2012.

[4] S. A. Cook, "The complexity of theorem-proving procedures," ser. STOC '71. ACM, 1971.

[5] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T)," 2006.

[6] A. Bauer, M. Pister, and M. Tautschnig, "Tool-support for the analysis of hybrid systems and models," in *Design, Automation and Test in Europe (DATE)*, 2007.

[7] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver." Springer, 2008.

[8] M. Fränzle, C. Herde, *et al.*, "Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure," *Journal on Satisfiability, Boolean Modeling and Computation*, 2007.

[9] M. Davis and H. Putnam, "A computing procedure for quantification theory," 1960.

[10] N. Een and N. Sörensson, "MiniSat v1.13 - A SAT Solver with Conflict-Clause Minimization, System description for the SAT competition," 2005.

[11] P. Doherty, J. Kvarnström, and A. Szalas, "Temporal composite actions with constraints," in *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2012.

[12] P. Doherty and J. Kvarnström, "Temporal Action Logics," in *Handbook of Knowledge Representation*. Elsevier, 2009.

[13] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," in *ANNUAL ACM IEEE DESIGN AUTOMATION CONFERENCE*. ACM, 2001.

[14] J. a. P. M. Silva and K. A. Sakallah, "Grasp – a new search algorithm for satisfiability," ser. ICCAD '96. IEEE Computer Society.

[15] M. Fränzle and C. Herde, "Hysat: An efficient proof engine for bounded model checking of hybrid systems." *Formal Methods in System Design*, 2007.

[16] M. Riedmiller and H. Braun, "Rprop - a fast adaptive learning algorithm," *International Symposium on Computer and Information Sciences - ISCIS*, 1992.

[17] A. Shtof, A. Agathos, Y. Gingold, A. Shamir, and D. Cohen-Or, "Geosemantic Snapping for Sketch-Based Modeling," *Computer Graphics Forum*, vol. 32, no. 2, pp. 245–253, 2013.

[18] H. Skubch, "Modelling and Controlling of Behaviour for Autonomous Mobile Robots," Ph.D. dissertation, University of Kassel, 2012.

[19] Y. Shang, M. P. Fromherz, and L. Crawford, "A new constraint test-case generator and the importance of hybrid optimizers," *European Journal of Operational Research*, 2006.

[20] J. M. Crawford and L. D. Auton, "Experimental results on the crossover point in random 3-sat," *Artificial Intelligence*, 1996.

[21] A. Petcu, "A Class of Algorithms for Distributed Constraint Optimization," Ph.D. dissertation, Swiss Federal Institute of Technology (EPFL), 2007.