Combining Task and Motion Planning is Not Always a Good Idea

Fabien Lagriffoul, Lars Karlsson, Julien Bidot, Alessandro Saffiotti
Örebro University
Email: name.surname@oru.se

***Abstract*—Combining task and motion planning requires to interleave causal and geometric reasoning, in order to guarantee the plan to be executable in the real world. The resulting search space, which is the cross product of the symbolic search space and the geometric search space, is huge. Systematically calling a geometric reasoner while evaluating symbolic actions is costly. On the other hand, geometric reasoning can prune out large parts of this search space if geometrically infeasible actions are detected early. Hence, we hypothesized the existence of a search depth level, until which geometric reasoning can be interleaved with symbolic reasoning with tractable combinatorial explosion, while keeping the benefits of this pruning. In this paper, we propose a simple model that proves the existence of such search depth level, and validate it empirically through experiments in simulation.**

I. Introduction and Related Work

The first attempts to endow robots with planning capabilities (e.g., [11]) were based on hierarchical architectures, with typically three levels of abstraction: a deliberative layer on top, a path planning layer, and a control layer for execution at the bottom. When robot platforms became more complex, these architectures became obsolete, because the deliberative layer could not reason about all the consequences of abstract actions in the real world. In other words, a plan which is logically sound may not be executable on the real robot because geometric or dynamic aspects are not taken into account at deliberative time.

In order to address this problem, a new paradigm emerged, in which task and motion planning are not decoupled, but combined. Recently, several approaches for combining task and motion planning (CTAMP) were proposed. Among these approaches, different philosophies can be distinguished, depending on how the symbolic level and the geometric level interact. In the first type of approach, the motion planner steers the search process. In SamplSGD [13] for instance, the planner mainly works on a path planning problem, while the symbolic domain is used to structure the path planning problem and determine where to direct the search. In aSyMov [2], the planner uses cost functions to determine whether to invest time in expanding a set of probabilistic roadmaps for motion planning, or to call a task planner for heuristic evaluation of the next action. In the second type of approach, the task planner is steering the search, while a geometric reasoner is called to geometrically evaluate the preconditions and effects of symbolic actions. These approaches include SAHTN [15], semantic attachments [3], and [6], [7], [8]. A third type of approach consists in stating the symbolic planning problem in terms of logic programming. The failures which occur at the geometric level can thus be fed back as constraints into the logic program, and the causal reasoner can use this information to find alternative plans. The advantage of this approach is the expressiveness of the formalisms used (e.g., action language $C+$ in [4], ASP program in [1]) which can handle concurrency, non-deterministic effects, ramification, etc. The work presented in this paper is heading in the same direction as [5] and [14]. In these papers, the authors systematically compare different ways of integrating task and motion planning, e.g., hierarchical, interleaved, or re-planning. The spirit of this paper is the same, but instead of comparing extremely different algorithms, we studied how the exploration of the search space is affected with respect to *gradually* changing the level of coupling between task and motion planning.

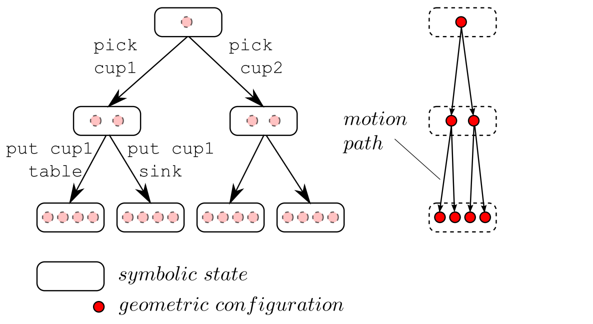

Fig. 1. Two nested search spaces: the symbolic search space(left), and a geometric search space nested in each branch of the symbolic search space(right).

In the present work, the task planner is steering the search while a geometric reasoner is called to evaluate the feasibility of symbolic actions. In this type of approach, two levels of complexity are nested within each other. The first level has to do with task planning. The nodes of this search space are the symbolic states, which are the result of applying symbolic actions to the previous symbolic states (see Fig. 1, left). The second level is geometric. The role of the geometric reasoner is to find a geometric instance for each symbolic state (a geometric configuration) and for each symbolic action (a motion path). This is not a trivial problem because a symbolic state can be instantiated into many geometric configurations, and there may be geometric dependencies between actions. Consider for instance the symbolic action `place cup1 table`. Different positions on the table and different orientations for the cup can be chosen, and these choices may prevent a later action to be executed: the cup may be in a position where it is not reachable by another robot, or it may be placed such that it prevents another object to be manipulated. In order to cope with such situations, and ensure completeness, a *geometric backtracking* mechanism is needed. Because of this, a geometric search problem is nested in each branch of the symbolic search tree (Fig. 1, right). The complexity of this sub-problem is exponential in the number of symbolic actions in the branch, and its branching factor depends on the resolution used for instantiating symbolic actions, i.e., for the `place cup1 table` action for instance, how many positions are sampled to place the cup on the table. *Geometric backtracking* occurs for action sequences which contain geometric dependencies, and for action sequences which are not feasible. More detail about *geometric backtracking* can be found in [9].

Because of this nested structure, the combined search space (symbolic $\times$ geometric) grows double-exponentially in the depth of search, which makes even simple problems intractable. Consider also that geometric reasoning is computationally expensive because assessing the reachability of a geometric configuration requires to call a motion planner. Hence, one may be tempted to perform pure symbolic search until a symbolic plan is found, and *then* search for a geometric solution. This approach works well only if symbolic and geometric levels can be decoupled, otherwise a lot of backtracking is necessary at the symbolic level, and the geometric reasoner is repeatedly called to evaluate similar actions sequences. It is important to notice that despite its cost, geometric reasoning is useful because it can prune the combined search space. Consider for instance the action `pick cup2` in Fig. 1. If this action turns out to be geometrically infeasible, half of the search space can be pruned out. The pruning is all the more effective as it occurs higher in the search tree. We hypothesize a geometric search depth, *up to which* geometric reasoning and symbolic reasoning can be combined in order to take advantage of pruning, but *beyond which* search should only be done at the symbolic level to avoid intractable double-exponential growth. We refer to this scheme as LD-CTAMP (Limited-Depth Combined Task and Motion Planning).

In the next section, we describe an algorithm with does LD-CTAMP, and use a simplified model to analyze its properties. In section III, we experimentally evaluate this algorithm in a scenario of object manipulation by a humanoid robot. A short discussion follows in section IV.

II. A SIMPLE MODEL FOR LIMITED-DEPTH CTAMP

In this section, a model of the search process is proposed, which is simple enough to derive an analytical formulation, whilst conserving its basic properties. The aim is to get an order of magnitude of $\mathcal{N}$, the total number of nodes visited (symbolic + geometric) with respect to the geometric search depth D.

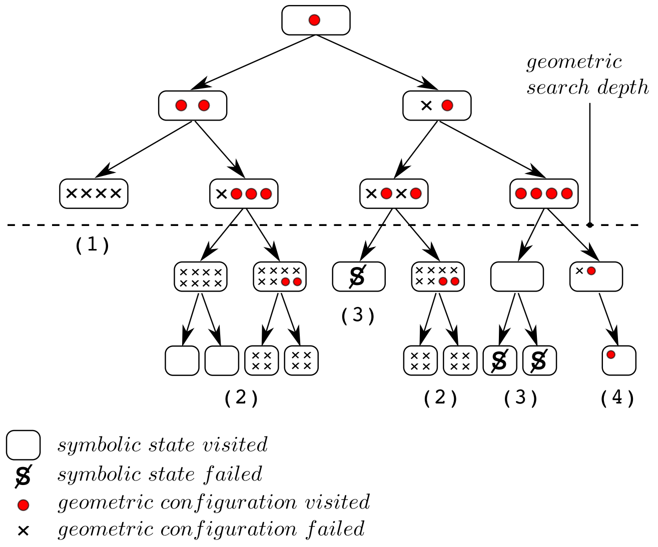

Fig. 2. Different cases occuring during search with LD-CTAMP.

A. LD-CTAMP

The main features of LD-CTAMP are summarized in Fig. 2. An important parameter of the model is the geometric search depth D, represented by the dashed line. Above this line, full CTAMP search is performed, i.e., each symbolic action which is applicable is geometrically evaluated. During geometric evaluation, it may happen that no geometric configuration is found to instantiate the symbolic action (1). This can occur due to collisions, inverse kinematic problems, or failure of motion planning. In this case, geometric backtracking occurs (previous geometric choices are reconsidered). If geometric backtracking fails, symbolic backtracking occurs, i.e., another symbolic action sequence is considered.

When search takes place *beyond* the geometric search depth, geometric reasoning is deactivated. Only when a symbolic goal-state is reached, geometric reasoning is done in order to geometrically instantiate the sequence of symbolic actions that led to this goal-state. The geometric evaluation may fail at different levels (2) or reach a solution (4). It may also fail at the symbolic level (3), in which case no geometric reasoning is needed.

B. Simplified model

We denote by N the depth of the goal state, and assume that the search front is on average located at this depth (see Fig. 3). We assume $N \geq D$. We assume an average branching factor g for the geometric search space. g reflects the resolution which is used by the geometric reasoner to sample geometric configurations. For the symbolic search space, we use two different branching factors:

- s' is used for the CTAMP search space (before D). It is less or equal to s because during CTAMP, geometric reasoning prunes out some symbolic actions, which decreases the effective branching factor.

- s is used for the "pure symbolic" search space (beyond D). This branching factor results from the symbolic planning domain (number of operators, number of symbols).

We denote by $\alpha = s'/s$ the *geometric success rate* of the problem. α tells how often on average symbolic actions are actually geometrically feasible. For instance, a low α value means that many symbolic actions turn out to be infeasible when they are evaluated geometrically. Note that $\alpha \in [0, 1]$, since $s' \leq s$. This parameter depends on many factors, which are discussed in the last section.

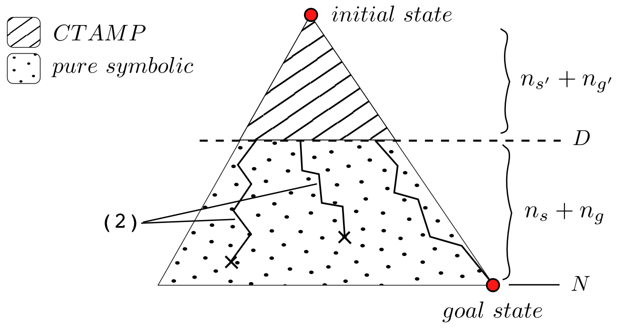

Fig. 3. Simplified representation of the search space.

For the rest of the calculations, the number of nodes visited at some depth d with some branching factor b is approximated by the number of leave nodes at this level, b^d. The total number of nodes visited by LD-CTAMP is

$$\mathcal{N} = n_{s'} + n_{g'} + n_s + n_g,$$

where $n_{s'}$ and $n_{g'}$ are respectively the number of symbolic and geometric nodes visited during CTAMP, and n_s and n_g the number of symbolic and geometric nodes visited during pure symbolic search (see Fig. 3). Lets us look at each term individually. The number of symbolic nodes visited during CTAMP is approximated by

$$n_{s'} \simeq s'^D.$$

$n_{g'}$ depends on the geometric branching factor g and depth D, but it is also proportional to the number of symbolic branches, because a geometric search problem is nested in each symbolic action sequence (see Fig. 1). Hence,

$$n_{g'} \simeq g^D s'^D.$$

In practice however, it is not possible to explore all the geometric search space. Remember that each time a geometric configuration is visited, a path planner is called to check the feasibility of the action, which takes tine. Therefore, a cutoff value G_{max} is set on the number of geometric configurations visited in each symbolic branch:

$$n_{g'} \simeq min(G_{max}, g^D)s'^D.$$

On average, the term $min(G_{max}, g^D)$ does not depend on D, so we approximate it by a constant term $G_{avg} \in [0, Gmax]$. Hence,

$$n_{g'} \simeq G_{avg} s'^D.$$

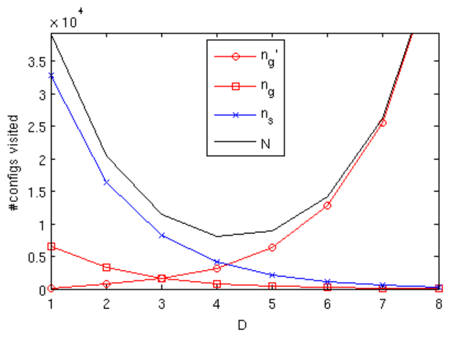

Fig. 4. Typical behavior for $G_{avg} = 200, s = 4, g = 20, N = 8, \alpha = 1/2$ and $\beta = 0.001$.

n_s is the number of symbolic nodes contained in the search tree which is the continuation of the CTAMP tree, but with a different branching factor, hence

$$n_s \simeq s'^D s^{N-D}$$
$$n_s \simeq s^N \alpha^D.$$

n_g depends on the branching factor g and the depth $N-D$, but furthermore on the number of successful symbolic branches (2). To represent these, we introduce another parameter $\beta \in [0, 1]$, the *symbolic success rate*, which is the ratio of the number of symbolic goal-states at depth N by the number of symbolic leave nodes at depth N. Hence, the number of successful symbolic branches is $n_s\beta$, and

$$n_g \simeq g^{N-D} n_s \beta.$$

But again, a cutoff value on g^{N-D} is needed:

$$n_g \simeq G_{avg} n_s \beta$$
$$n_g \simeq G_{avg} s^N \alpha^D \beta.$$

We notice that $n_{s'} \ll n_{g'}$, so we can simplify

$$\mathcal{N} \simeq n_{g'} + n_g + n_s$$
$$\mathcal{N} \simeq G_{avg} s'^D + s^N (1 + G_{avg}\beta)\alpha^D.$$

C. Analysis of the model

$\mathcal{N}$ is the positive weighted sum of two exponential functions of D. s' is assumed to be greater than 1, so $n_{g'}$ is an increasing exponential, while since $\alpha < 1$, $n_g + n_s$ is a decreasing exponential. Consequently, for low values of D, $\mathcal{N}$ is dominated by $n_g + n_s$, whereas for high values of D, it is dominated by $n_{g'}$. Hence, as D increases, $\mathcal{N}$ decreases until a minimal value, and increases afterward, as illustrated in Fig. 4. We denote by D^* the optimal value (which minimizes $\mathcal{N}$) for D. According to the model, using the limited-depth CTAMP scheme is beneficial. As an example, with the parameter values used in Fig. 4, the minimal number of configurations visited is 8131 ($D^* = 4$), versus 51712 for $D = 8$ (fully combining task and motion planning).

The pattern observed in Fig. 4 is stable with respect to the parameters G_{avg}, s, g, N, β. The parameter that critically affects the behavior of the model is α, the *geometric success*

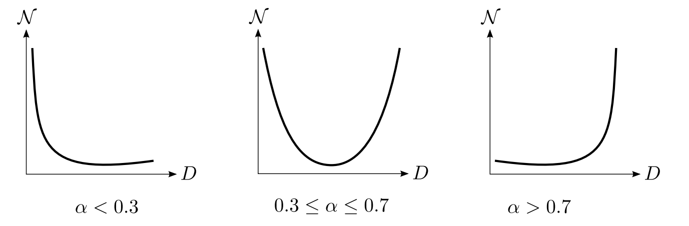

Fig. 5. Different behaviors observed for $\mathcal{N}(D)$ depending on the *geometric success rate* α.

rate (see Fig. 5). When α tends towards 0 (meaning that few symbolic actions are geometrically feasible), then D^* tends towards infinity, suggesting that it is more effective to do CTAMP all the way. Conversely, when α tends towards 1 (most symbolic actions are geometrically feasible), then D^* tends towards 0, which corresponds to a full decoupling of symbolic and geometric reasoning.

For taking advantage of LD-CTAMP, the difficulty is to correctly evaluate the parameter α, in order to choose the appropriate value for D. We chose the value $\alpha = 1/2$ because we work in the domain of object manipulation by a humanoid robot. In this domain, half of the actions are performed with the left arm, and half with the right arm. The objects which are in the middle of the workspace are reachable by both arms, but often, objects are only reachable by one of the arms, which suggests that around half of the symbolic actions may be pruned by geometric reasoning. Collisions and kinematic constraints also have a negative effect on the success of some actions, but geometric backtracking often copes with that. In the next section, this simple model is evaluated against experiments conducted in a simulation.

III. Experimental evaluation

We used the simulation environment provided by DLR[1] for the robotic platform Justin [12]. Justin is a humanoid robot with two arms with 7 DoF each. In the proposed scenario, the robot is situated in front of a table, on which some objects have been placed. Two experiments were performed, in which the task was to place respectively two and three cups in the box. The cup in the right-most pose (Fig. 6) has to be handed over to be placed in the box, which requires 4 actions. The other cups can be directly placed in the box with the right arm in two actions.

For task planning, we used the planner JShop2 [10], which was modified so that all the features described in section II-A were implemented. The domain consists of 5 operators: `pick`, `place`, `pick_regrasp`, `place_regrasp` (for hand-overs) and `move_away` (to place the arm in a resting position). The operators have different parameters that specify which arm and which type of grasp is used, e.g., `pick right top cup1` means that `cup1` is grasped with the right arm, using a top-grasp. We wrote 4 `move_object` methods, which implement different ways of moving an object to a given location:

[1]Deutsche Zentrum für Luft und Raumfahrt

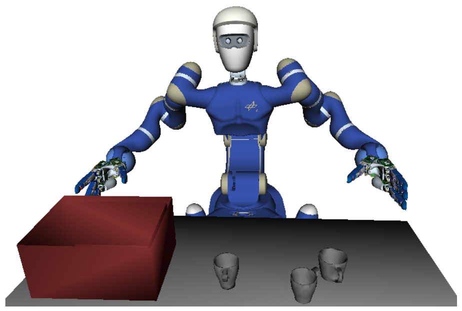

Fig. 6. Experimental setup: an object manipulation scenario with a simulation of the robotic platform Justin.

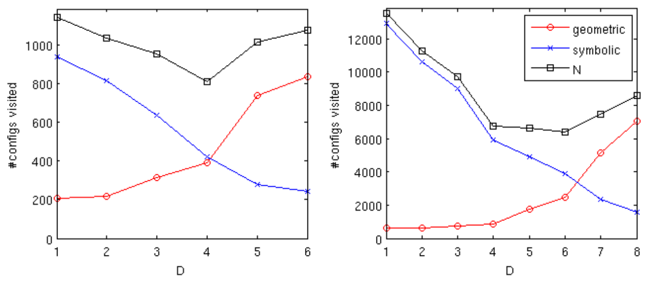

Fig. 7. Results for the experiment with 2 cups (left) and 3 cups (right). The total number of geometric nodes, symbolic nodes, and $\mathcal{N}$ are represented.

- pick and place the object;
- hand-over (pick, place_regrasp, pick_regrasp, place);
- move a potential obstacle, then move the object;
- pick the object, move the opposite arm away, and place the object.

All together, these four methods allow to cope with most object manipulation problems.

The geometric reasoning component is more complex. It tries to find a geometric instance for each symbolic action added to the current plan. This requires to sample several geometric configurations (e.g., for a `place` action, several positions are possible for the object) until one is found which is collision-free and kinematically feasible. Then, a motion planner is called to check if this configuration is reachable from the previous configuration. If this process fails, a backtracking mechanism is triggered in order to reconsider the geometric configurations of the previous actions, because they may be the cause of failure (see [8] for more details). The number of geometric configurations visited during geometric backtracking was set a cutoff value (G_{max}) of 200.

We counted the number of symbolic states and the number of geometric configurations visited for different values of D, until the first solution was found. The results (Fig. 7) show that the real algorithm follows the behavior predicted by the model, despite the approximations that were used. An important result which does not appear in these charts is the planning time. In order to keep the model simple, only the number of nodes visited was considered in the model. But in practice, visiting a geometric configuration takes much more time than visiting a symbolic state (because of motion planning). Considering time instead of number of configurations visited would obviously lead to a lower optimal geometric depth D^*.

IV. Discussion

We proposed a simple model which predicts the existence of a trade-off between the number of symbolic states visited and the number of geometric configurations visited in CTAMP problems. According to this model, the minimum planning effort in a CTAMP problem is achieved by performing interleaved symbolic and geometric search until a given level D^*, and performing only symbolic search (with a final geometric check) after that depth. We have conducted some experiments in simulation that validate this model in a simple scenario.

As a continuation of this work, the first idea that comes to mind is to augment LD-CTAMP with a function that counts the number of symbolic actions pruned out by the geometric reasoner, in order to compute an estimation of the *geometric success rate*, and automatically tune the geometric search depth to its optimal value. However, the aim of this work is not to propose a new CTAMP algorithm, but rather to get a better understanding of the search space in combining task and motion planning problems.

Thanks to this work, we understood that our current approach (fully combining task and motion planning, see [8]) worked well because we intentionally constructed intricate problems to evaluate it, i.e., problems with a low *geometric success rate*. This type of problems benefits from the strategy of fully combining task and motion planning, but both the model and the experiments show that this strategy is not the best one in the general case. According to the model, scenarios with a higher *geometric success rate* are intractable with this type of approach, because the search effort is wasted in geometric evaluation of symbolic sequences which do not lead to a symbolic goal-state.

Finally, this work opens a possibly different line of approach for CTAMP problems. What is suggested by our results is that the characteristics of the problem have a significant impact on the efficiency of the search algorithm. Therefore, instead of searching for *the* best algorithm capable of combining task and motion planning, it may be worth looking into how to classify CTAMP problems according to their intrinsic properties, and adapt the search algorithm accordingly. We identified the *geometric success rate* as one of these properties, but this concept is hard to define. For sure, it is related to the kinematics of the robotic platform and the clutteredness of the workspace, which have a direct effect on the success of geometric actions. But sometimes, the *geometric success rate* is the result of a complex interaction between the symbolic level and the geometric level. For instance in our scenario, placing *one* cup in the box is not problematic per se, but placing three or more cups definitely reduces the chances of success. On the other hand, when *stacking* cups on top of each other, the number of cups does not really affect the chances of success. More investigation is required to understand this type of interactions and make the concept more accurate.

Acknowledgments

This work was partially supported by EU FP7 project "Generalizing Robot Manipulation Tasks" (GeRT, contract number 248273). We would like to thank in particular Florian Schmidt and Daniel Leidner from the Robotics and Mechatronics Center of DLR, for their advice, and for the functionalities they developed in Justin's simulator, which made this work possible.

References

[1] Erdi Aker, Volkan Patoglu, and Esra Erdem. Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution. *Intelligent Service Robotics*, 5(4):275–291, 2012.

[2] Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *Int. J. Rob. Res.*, 28(1):104–126, 2009. ISSN 0278-3649.

[3] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. Semantic attachments for domain-independent planning systems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS09)*, pages 114–122, Thessaloniki, Greece, 2009.

[4] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4575–4581, 2011.

[5] J. Guitton and J.-L. Farges. Towards a hybridization of task and motion planning for robotic architecture. In *International workshop on Hybrid Control of Autonomous Systems (HYCAS)*, pages 21–24, 2009.

[6] Julien Guitton and Jean-Loup Farges. Taking into account geometric constraints for task-oriented motion planning. In *Proc. Bridging the gap Between Task And Motion Planning, BTAMP'09 (ICAPS Workshop)*, 2009.

[7] Leslie Pack Kaelbling and T. Lozano-Perez. Hierarchical planning in the now. In *Proc. of Workshop on Bridging the Gap between Task and Motion Planning (AAAI)*, 2010.

[8] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt. Combining task and path planning for a humanoid two-arm robotic system. In *Proceedings of TAMPRA: Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*, 2012.

[9] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson. Constraint propagation on interval bounds for dealing with geometric backtracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 957–964, 2012.

[10] Dana S. Nau, Héctor M. Avila, Yue Cao, Amnon Lotem, and Steven Mitchell. Total-Order Planning with Partially Ordered Subtasks. In *Proceedings of the International*

Joint Conference on Artificial Intelligence (IJCAI), pages 425–430, 2001.

[11] Nils J. Nilsson. Shakey the robot. Technical report.

[12] Ch. Ott, O. Eiberger, W. Friedl, B. Bäuml, U. Hillenbr, Ch. Borst, A. Albu-schäffer, B. Brunner, H. Hirschmüller, S. Kielhöfer, R. Konietschke, T. Wimböck, F. Zacharias, and G. Hirzinger. A humanoid two-arm system for dexterous manipulation. In *2006 IEEE Int. Conf. on Humanoid Robots*, pages 276–283, 2006.

[13] E Plaku and G Hager. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *Proceedings of ICRA10*, 2010.

[14] P. Schüller, V. Patoglu, and E. Erdem. A systematic analysis of levels of integration between low-level reasoning and task planning. In *Workshop on Combining Task and Motion Planning (IEEE International Conference on Robotics and Automation)*, 2013.

[15] Jason Wolfe, Bhaskara Marthi, and Stuart J. Russell. Combined task and motion planning for mobile manipulation. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS10)*, pages 254–258, 2010.