Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search.

Mamoun Gharbi, Raphaël Lallement, Rachid Alami

To cite this version:

Mamoun Gharbi, Raphaël Lallement, Rachid Alami. Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search.. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep 2015, Hamburg, Germany. hal-01955200

HAL Id: hal-01955200
https://laas.hal.science/hal-01955200

Submitted on 14 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Combining Symbolic and Geometric Planning to synthesize human-aware plans: toward more efficient combined search.

Mamoun Gharbi[1,2], Raphaël Lallement[1,2] and Rachid Alami[1,3]

***Abstract*— We are combining symbolic and geometric planning to synthesize human-aware plans in order to deal with the complex and highly intricate planning problems induced by Human-Robot collaborative object manipulation.**

In this paper, we summarize our previous contributions – refining symbolic actions at geometric level, during the symbolic planning, in order to assess their feasibility and computing the geometric side effects–, then we present the current contributions meant to tighten the cooperation between the symbolic planning and the geometric planning: the symbolic planner helps the geometric one by providing it with constraints and domain-expert knowledge making the geometric planner more efficient, and the geometric planner helps the symbolic one to find the best plan based on social costs computed at geometric level.

We also propose different examples, highlighting the interest of such cooperation between the planners in simulation and on our PR2 robot.

I. INTRODUCTION

We are developing a symbolic-geometric planner in the context of fetch & carry and collaborative human-robot object manipulation [1]. Besides, we would like the planner to be used in the context of situated dialogue [2], [3]. We argue that such a context is very challenging and opens interesting and "subtle" issues to the Combined Task and Motion Planning problems.

In such a context and perhaps more than in standard robotic manipulation problems (the block world problem and its *robotics* variants, for instance), there is a need for a more sophisticated reasoning since actions and motions are far more intricate, and the contexts in which they are performed, as well as how they have to be performed, can induce complex and highly interdependent decisions.

The planner should be able to synthesize plans for every agent, while taking into account the intricacy induced by having them sharing the same space and task. One difficulty comes from our intention to reason and consider affordances and perspective taking from both human and robot sides and, based on this, to produce collaborative actions where every agent has to act.

Fig. 1 shows an example of such a problem, where a robot needs to take decisions and computes plans in a human populated space. The bottom part corresponds to a plan automatically synthesised by our planner (containing both the action sequence for every agent and the geometric information needed, such as the trajectories, the grasps, and the object placements). The plan takes also into account a number of human-aware constraints such as minimising the humans efforts or avoiding to disturb them.

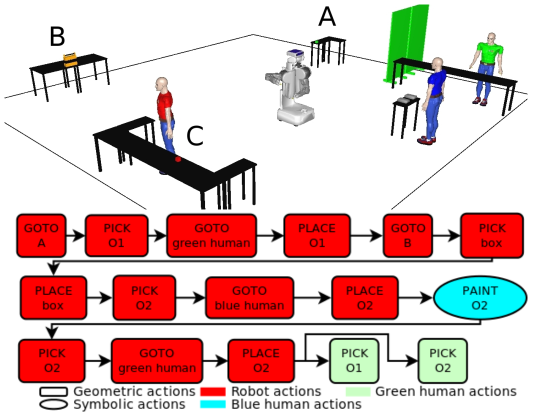

Fig. 1. The robot needs to bring two processed objects to the green human. A processed object O1 is available at A, and two unprocessed objects O2 and O3 are available at B and C (the blue human can process them). To pick O2 the robot needs to move out the orange box, and to reach O3 the robot needs to navigate behind the red human. Our planner is able to find a solution to this problem while taking into account humans preferences, affordances and comfort. The bottom figure shows the solution plan.

In a multi-agent, human-aware context, the action effects and their costs cannot be computed at symbolic level, especially while taking into account the human comfort and preferences [4]: the interleaving between symbolic and geometric planning becomes mandatory.

This paper reports on the extensions of a planner we have already presented in [5], [6]. We provide details on how the ramification problem is tackled and present a number of extensions specially dedicated to the improvement of the plan search: (1) a set of high level geometric constraints, (2) the ability to encode domain-expert knowledge at both levels, (3) well-informed cost estimation of actions and (4) the ability of the geometric planner to provide several instantiations of the same action. This permits to address more challenging problems and to integrate human-aware planning considerations.

II. RELATED WORK

Merging symbolic and geometric planning is a growing field that has been a focus to a number of researchers over the few last years. Various approaches are pursued, some are similar to our previous work, such as [7] where the

*This work was conducted within the EU SAPHARI project funded by the E.C. division FP7-IST under contract ICT-287513.

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France `firstname.lastname at laas.fr`

[2]Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

[3]Univ de Toulouse, LAAS, F-31400 Toulouse, France

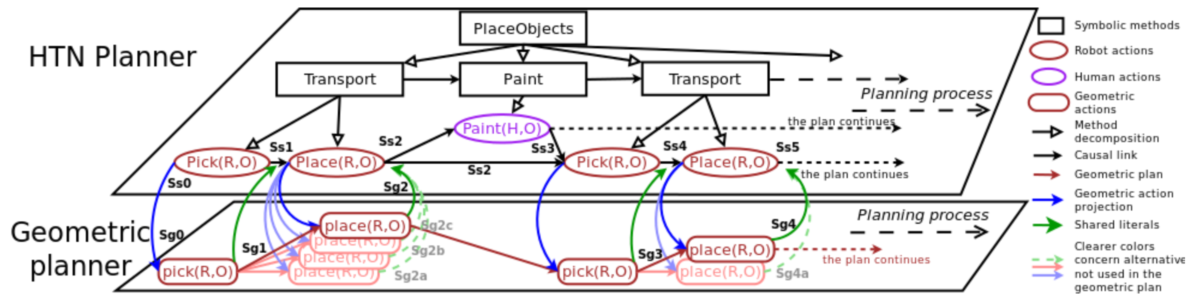

Fig. 2. A snapshot of the planner during the planning process. This plan is a "multi-stream" plan: human and robot actions. The robot has to place an object so the human can paint it and then the robot will transfer it to another place. The plan is built incrementally: the symbolic plan is maintained by the HTN planner while its geometric counterpart –the geometric plan–, the geometric refinement of some actions (the others are purely symbolic), is maintained by the geometric planner. When an action preconditions are not respected (*Paint* here), the HTN planner backtracks to the previous backtracking point, (the refinement of *Place* here) and tries alternatives to this action (the different refinements of *Place* in the geometric planner layer). The preconditions can be purely symbolic or computed by the geometry (predicates such as is on, is Reachable by) which enable us to tackle the ramification problem. The planner tries 3 different action instantiations (called alternatives) before finding the right object placement that will allow the human to paint it.

authors guide the search at geometric level with the symbolic planning. Others focus on different aspects, such as [8], where the authors construct a graph (the nodes are robots positions linked to feasible actions at task level, and the edges are collision free paths) and use a Satisfiability Modulo Theories to find a path in it. Similarly, [9] also builds a graph representing different configurations reachable by the robot and the FastForward algorithm is used to find a solution.

In [10] and [11], the authors use *semantic attachments*, which are external procedures called by the task planner to assess the feasibility of an action at the geometric level (e.g. if a collision free path exists between two positions). [12] extends this approach by adding a geometric backtracking: in order to assess an action feasibility, the geometric planner can change the choices made for the previous actions without acknowledging the symbolic planner.

In [13], the authors use a goal driven task planner that enables calls to external procedures during the planning phase. In case of failure, this procedure provides a reason to the planner which enables it to find an alternative plan. Using failure reasoning to guide task planning is also stressed out in [14] where a causal reasoner is used to find a plan, then a motion planner tries to solve the underlying motion problem, and if the latter fails, a constraint is added to the symbolic domain and the causal reasoner tries to find a new plan. [15] uses off-the-shelf planners and formulates a representational abstraction enabling to combine them: each action in the symbolic plan can have multiple instantiations at geometric level corresponding to the different geometrical choices.

The authors in [16] compute a task-plan, and use interval bounds which are constraints on object/robot positions to prune out geometric choices. Another approach, proposed in [17], is to formalise the problem as geometric constraints between agents and objects, and to use a Constraint Satisfaction Problem solver to find the solution.

III. SYMBOLIC-GEOMETRIC PLANNING PROBLEM

We define a problem combining the symbolic and geometric planning as the 6-tuple $\langle D_s, D_g, S_{s0}, S_{g0}, g, E\rangle$ where D_s stands for the symbolic domain, containing all the symbolic tasks, while D_g is the geometric domain that contains all the geometric actions. S_{s0} and S_{g0} are respectively the symbolic and the geometric initial states (they represent the SAME world state). g refers to the goal to achieve and E represents all the elements of the environment: the agents, the pieces of furniture, the objects and so on; they are called entities. The agents are treated as first-order entities since they correspond to robots and humans for which it will be necessary to compute motions. Entities are referred to, on both systems, with the same name to keep the correspondence. Their representation changes depending on the planner: at the symbolic level, attributes are associated to an entity in the form of predicates, whereas at the geometric level the same entity is described by its shape and its configuration space.

The geometric reasoning system is able to compute spatial relations between entities, such as between objects (in, on-top-of, ...) or between agents and objects (visibility, reachability, ...) [18]. This can be related to well known need to deal with the anchoring problem in order to fill the gap between the levels [19]. Those relations are named "Shared Literals" (as in [6]) since they are generated by the geometric level and exploited by the symbolic planner.

The solution to this kind of problem is a set of feasible actions, where an action is defined by the agents and their motions. The different actions are sequenced thanks to causal links –computed at the symbolic level from the HTN hierarchy– forming the plan.

IV. ALGORITHMS AND EXTENSIONS

A. *Previous algorithms*

We have proposed various contributions to the problem of combining symbolic and geometric reasoning in order to produce pertinent and feasible robot plans.

In Asymov [20], [21] we essentially proposed a principled way to link the two planners thanks to a geometric level able to tackle the so-called "manipulation planning problem" [22] and that allows to explicitly take into account the topological changes occurring in the configuration space, when a robot grabs or releases an object. Asymov provided a well founded translation of pick and place actions (and similar actions) into 'transit' and 'transfer' motion planning requests even in multi-object and multi-robot contexts.

More recently we focused on a complementary approach [5], [6]: exploiting the capacity of the Hierarchical Task Network (HTN) [23] techniques to encode domain knowledge and developing a geometric planner capable of planning actions with several levels of abstractions, opening to more elaborate action instantiations. Such a combination provides several key features: a clean interface which corresponds to the anchoring problem and allows to better exhibit and master the links between the incremental processes of producing the symbolic plan and its geometric counterpart.

In [5] we presented a geometric backtracking algorithm which allows to reconsider the previous geometric choices by trying various alternatives to the previously computed actions and tests the validity of these alternatives by computing their geometric effects. In [6], the approach is different since the symbolic planner creates multiple instances for the same action (if a geometric refinement is needed) and backtracks on this instances, which are in fact, different geometric alternatives for the same symbolic action. If an actions is successfully refined, geometric effects, the shared literals, are computed.

The algorithm we use in this paper is presented in details in [5] and illustrated in fig. 2. The implementation is different: previously the link between the symbolic planner and the geometric planner was encoded directly into the symbolic domain, now a complementary module handles this link. At the geometric level, the implementation is more generic, and is able to tackle more complex problems linked to the symbolic-geometric planning problem.

B. *Algorithm extension*

The ramification problem occurs when all the effects of an action cannot be determined beforehand. When a motion is involved in an action, the symbolic planner cannot compute all the effects: a too-complex world model would be required. Furthermore, computing the human affordances (objects reachability, visibility and so on) makes it even more complex. Fig. 3 shows an example of this problem: the robot needs to place three objects on the table in front of it in order for the human to be able to reach the three of them at the same time. In the figure 3-C the robot places the third object, but this makes the first one no longer reachable.

In order to (partially) tackle this problem, we use the shared literals: after a geometric action is planned, we compute those literals for the new end state and send them to the symbolic layer. If some of the literals prevent a further action preconditions to apply, a backtrack is triggered and another geometric solution is requested. This process goes on until a valid plan is found, a given maximum number of geometric solutions (given by the domain expert) are tested, or no other geometric solution is available.

The problem is only partially tackled due to the discrete set of shared literals the system is able to compute: if a shared literals does not exist, the problem will not be tackled.

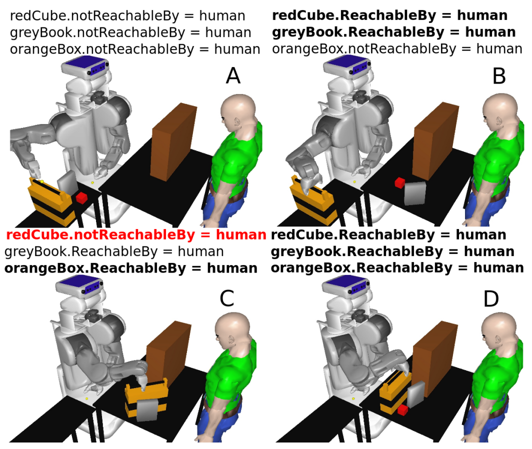
redCube.notReachableBy = human
greyBook.notReachableBy = human
orangeBox.notReachableBy = human
A
redCube.ReachableBy = human
greyBook.ReachableBy = human
orangeBox.notReachableBy = human
B
redCube.notReachableBy = human
greyBook.ReachableBy = human
orangeBox.ReachableBy = human
C
redCube.ReachableBy = human
greyBook.ReachableBy = human
orangeBox.ReachableBy = human
D

Fig. 3. The robot can reach the three objects (the red cube, the grey book and the orange box) and needs to place them on the table, such as the three of them are reachable by the human at the same time. A is an initial geometric situation, B is a step of the planning process where the robot has already placed the red cube and the grey book reachable to the human. In C, the robot places the orange box reachable to the human, and by doing so, obstructs the human reachability to the red cube (this is computed by the geometric reasoner). Finally D shows the result of a plan found after backtracking on a number of actions. This illustrates the ramification problem.

The symbolic layer exploits the shared literals when checking the preconditions, such as most other planners of the literature. In addition to the precondition checks, we added a "goal" for the abstract tasks (methods) under the form of literals. This goal is specified in the domain and used to check if the method decomposition has effectively reached the target goal. If the goal is achieved, the resolution continues, otherwise a backtrack is triggered. This mechanism is especially useful when, during a method decomposition, one or multiple actions can break previously achieved sub-goals in this same decomposition.

The shared literals are also used as constraints set by the symbolic layer when refining an action in the geometry. Then the constraints serve as supplementary goals to the action and drive the geometric search since they forbid some solutions. A constraint is a literal that must be true in the action geometric end state, for instance $Object.ReachableBy == Human$.

To better understand how constraints might be efficient in our context, a little reminder of the algorithm to find an action solution is needed: first it finds a position for the object, then, it finds a configuration for the agent, using inverse kinematic, and finally, it looks for a trajectory between the agent current state and this configuration.

For the constraints to be useful, they are tested as soon as possible: if it is constraints over the object position, they are tested before calling the inverse kinematic, if it is about the agent configuration, they are tested before the motion planning. Using the constraints enables to save computation time as they are tested before the computation is finished.

This constraints, when added to the symbolic domain, consist on a domain knowledge able to drive the geometric search. Another useful domain knowledge is the virtual

TABLE I

RESULTS FOR THE PICK & PLACE EXAMPLE.

30 runs(time in s)	*Place*	*PlaceR*	*PlaceRGoal*	*PlaceRC*
Mean time(stdev)	44.3(6.1)	27(3.6)	23(2.7)	16.2(1.8)
Nb tasks tried	67.5	11.6	9.7	8.6
Nb alternatives	59.6	3.8	2.5	1.6

actions. Virtual actions are simple, fast computing actions, used to test if a future action might be possible. Those actions do not ensure the infeasibility of the future actions, but, if they succeed, ensure their feasibility.

The example presented later in this paper (subsection V-B) concerns a **virtualPlace**[1]. This action places a virtual object[2] on a support to test if it fits. In this case, any items this virtual object may contain will fit on the support.

The last contribution of this paper, is the cost usage: the symbolic planner uses cost-driven search [24], moreover each action has a cost function given by the domain expert. However, the geometric planner as it computes the actual trajectory, has a better cost estimation and can provide the symbolic planner with precise costs representing different parameters such as the energy needed, the time to execute or social rules (e.g. avoiding navigation behind humans [25]).

V. EXAMPLES & RESULTS

In this section, we present different examples involving at least a human and a robot (the PR2), where a standard two-layer planning architecture would not be able to solve the problem or would not find the best solution. The experiments were run on a quad-core Intel Core i7 processor and 8GB of RAM, running Ubuntu 14.04.

A. Exploiting reachability computation to enrich reasoning about pick & place

This example shows the interest of having a backtracking algorithm, different levels of abstraction in the geometric planner, constraints, symbolic goals and how our system handles the ramification problem.

Fig. 3-A shows a geometric initial situation S_{g0}. The robot needs to place the three objects (the red cube, the grey book and the orange box) on the table in front of it, reachable by the human (at the same time, in the end of the task).

The shared literals used in this example are $ReachableBy$ (which objects are reachable to each agent) and $IsOn$ (which object is on which one). The available geometric actions in D_g are **pick**, **place**, **placeR** and **placeRC**. **placeR** is an action where the robot places an object $ReachableBy$ the target agent (the human here). **placeRC** is similar to it but adds constraints to the action: the constraints are for the objects already placed on the table to still be $ReachableBy$ the same target agent once the action is performed.

The highest-level task in the symbolic domain D_s is *PlaceObjects* and is composed of the succession of three

[1]*italic* is used for the symbolic layer, **bold** is used for the geometric layer.

[2]an object that can contain other objects, and that can be ignored by the collision checking when needed. The virtual objects are given to the system, each virtual object correspond to a set of objects it can contain.

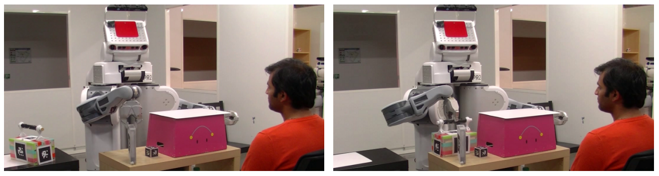

Fig. 4. The same experiment as in fig. 3 described in section V-A was held on the PR2 robot (the robot needs to place the three objects accessible to the human). The environment was set, then a plan (both at symbolic and geometric level) was computed by our algorithm and finally executed (the execution part is not a contribution of this paper).

Transport tasks followed by a *Validate* action. The *Transport* method contains a *Pick* action, then a *Place* and finally a *CheckReachable*. The goal of the latter is to check if the object involved in *Transport* is reachable by the human. The *Validate* action is similar but does the reachability test for all the objects the robot should place and is used as a goal test. The objects placement order is given to the planner.

There are four variants of *PlaceObjects*: the first is presented above, the second, *PlaceObjectsR*, where *PlaceR* is called instead of *Place*, the third, *PlaceObjectsRC*, uses *PlaceRC* instead of *Place*. The set of objects to specify in this function is the list of the previously placed objects, which prevent this action from breaking the predicates $Object.ReachableBy == targetAgent$ tested in *CheckReachable* and *Validate*. Finally the fourth, *PlaceObjectsRGoal*, which is similar to *PlaceObjectsR* (the second variant) but where the goal of each method is specified such that it must keep the previously-placed objects reachable in addition to make the newly-placed object reachable.

Table I shows a clear difference between the domains using *Place*, *PlaceR*, *PlaceRGoal* and *PlaceRC*. In the first case, the action *CheckReachable* preconditions are rarely met, resulting in a high number of alternatives computed and a long computation time (∼15min). When *PlaceR* is used, the number of alternatives computed decreases significantly: the geometric planner directly places the objects reachable by the human. Placing a new object may change the reachability of the previously placed objects, –the ramification problem depicted in fig. 3–, the constraints enforcement in *PlaceRC* prevents this behaviour, greatly reducing the number of alternatives needed and the computation time. *PlaceRGoal* gives better results than *PlaceR*, the ramification problem is detected sooner enabling a faster recovery: for instance when the second object (the grey book) is placed, if it breaks the reachability of the first one (red cube) the *PlaceR* variant will only detect it at the final *Validate* while the goal enforcement ensures an earlier detection. However *PlaceRC* gives better results as it prevents the ramification problem from occurring.

This example is based on backtracking, shared literals and abstract actions. [20] and [12] can reproduce the first results (**placeR** and **placeRC**) while [16] and [9], due to their constraint based approaches, can reproduce the latest ones (under the condition of adding the human-aware predicates).

This experiment (with slightly different items, but with the same problems) was held on the actual robot (PR2) as well,

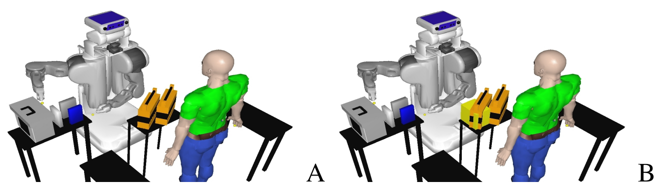

Fig. 5. The robot has to place the three books on the table in front of it. This table is cluttered by other objects, making it impossible to place more than one or two books. Case A shows an initial situation while B shows the **VirtualPlace** test made on the table to assess if there is enough space for placing all the books on the table.

fig. 4 shows key pictures of the execution and the attached video shows the whole plan execution.

B. Using domain-expert knowledge to drive the search

This example shows how it is possible using the combination of a symbolic and a geometric planner to implement a common-sense heuristic. When one tries to place several objects close one to the others, it tries to find a surface or to free a surface (if needed) where it is possible to put an "imaginary" (we call it virtual), big object that represents the volume to be occupied by the full set of objects.

Fig. 5-A shows an initial geometric states S_{g0}. The robot needs to place the three books in front of the human, but the table is cluttered with other objects. The available geometric actions in D_g are **pick** and **place**.

In this scenario the top-level task, *GiveCollection*, has two decompositions and, to decide which one to use, it tries a **virtualPlace** with a virtual object. If the virtual object can be placed, it means that there should be enough room to directly call the task *PutObjects*. This task is recursive and is called once for all objects in order to *Transport* them. (*Transport* is defined as the sequence of a *Pick* and a *Place* actions.) In the case where the virtual object can not be placed, the task *CleanTable* is called before using the *PutObjects*. *CleanTable* is also recursive and iterates through all objects on the goal table and stops when either the **virtualPlace** succeeds or all objects are removed. If the robot can not carry out this task (no space to place or no grasp for the objects) the human will help out.

In order to have a reference value, we have created a version of this domain where the **virtualPlace** is not used to choose whether to *CleanTable* or not, neither to stop the recursive call to *CleanTable* (which then stops only when all objects are moved away). Table II shows the two domains (with and without the virtual test). When **virtualPlace** is used, the planner tries to place a virtual object corresponding to the set of books, on the table (fig. 5-B) and if no space is found, calls *CleanTable* before *PutObjects*, otherwise it directly calls *PutObjects*. In the other case, the planner will try to place the books on the table, and will probably succeed to place one or two, but the space is limited, and the third book will not have enough space. Hence, the planner will backtrack several times over the *Transport* task before finally trying another decomposition (*CleanTable*), this implies to try several decompositions, refine and compute a lot of alternatives before finding the first valid plan. The significant time spent on useless backtracks can be seen in table II. On the other hand, with the **virtualPlace**, a first valid plan is found sooner since it realises that it is necessary to free the table before putting the objects, yet it may use few alternatives to correctly refine *CleanTable* or *PutObjects*.

TABLE II
RESULTS FOR THE FUNCTION **VIRTUALPLACE**

30 runs (time in s)	without *VirtualPlace*	with *VirtualPlace*
Mean time (stdev)	191.2 (8.6)	21.7 (1.45)
Nb tasks	162.6	25
Nb alternatives	62.9	0.3

In this example, the aim was to remove objects in the target area only when necessary, this has been of interest in [11], [13], [17].

C. Reconsidering the object choices based on the plan cost

This example shows how the geometry can guide and/or help the symbolic planner to find the best plan based on its quality. Number of research focuses on qualifying a geometric action, especially in a human-aware context. We use [25] in order to compute a cost for the robot navigation action, integrating the path length, the distance between the path and the humans and the length of the path passing closely behind the humans.

Fig. 6 shows two initial situations S_{g0} and their solutions (the blue lines). The difficulty consist on choosing the right object (between the two similar and available ones) to bring to the human. This choice should take into account the comfort of every human in the environment.

The available geometric actions in D_g are **pick**, **place** and **goto**, which is a navigation action where the search space is the set of positions within range of the target object.

The domain starts with the task *GiveObject* that randomly chooses an object: this is a backtracking point and since all the plans are computed in order to find the best, all objects are tested. It decomposes into *Take* and *MakeReachable* methods. The first method has two decompositions depending on whether the robot can *Pick* or has to first *GotoObject* before being able to *Pick*. The choice is based on a reachability test. The *MakeReachable* task again takes in account the reachability of the table to place the object on: it relies on *Place* but also has a second decomposition where it first uses *GotoTable* before the *Place*.

This example can be solved using a symbolic planner only, but our approach enables the planner to choose the best solution among all the possible plans while taking into account the geometric world and some specific social rules.

[15] describes an example close to this one, where the robot needs to choose between multiple symbolic plans and the best one based on a geometric computation of the cost.

Fig. 1 shows a variation of this example (The attached video shows a symbolic-geometric search in this domain, and the plan found for it), where there are more actions and interactions. The green human is waiting for two processed objects to be delivered, and there are two unprocessed objects

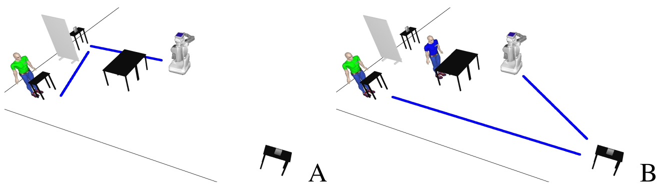

Fig. 6. The green human asks the robot to bring him an object. Two similar objects are available in the environment, and the robot needs to choose the one to bring to the human, depending on the environment (here the presence or not of the blue human). When the blue human is absent, the shortest path is chosen. When he is near the table, choosing this path makes the robot navigate behind him, thus the robot chooses the other object which is farther but, by doing this, it respects the human comfort.

and a processed one in the environment. The blue human is able to process the objects. The difficulties in this example are the followings: (1) to reach the unprocessed object at B, the robot needs to move the orange box, (2) to navigate to the unprocessed object at C, the robot goes behind the red human, (3) the planner needs to choose between the two unprocessed object based on cost evaluation, (4) the choice between the processed and unprocessed object can be done only by the symbolic planner.

The planner chooses for the robot to first bring the processed object to the green human (smaller number of actions), then, goes and get the unprocessed object at B after moving the orange box (as it is less disturbing for the red human) and brings it to the green human after the blue human has processed it.

VI. CONCLUSION

We have presented in this paper different improvements to a previously published algorithm, concerning the combination of symbolic and geometric planning in the context of human robot collaboration. We have provided details on how we handle the ramification problem by computing the actions side-effects and using the shared literals. We have also presented these improvements: (1) using shared literals as constraints to guide the geometric planner, (2) Adding requests to the geometric planner to asses the feasibility of future actions, (3) estimating accurately the action cost based on social rules and (4) using different alternatives of a geometric action for the same symbolic task.

In order to support these improvements, we implemented them on pick & place examples, however they can be used in other domains. In order to achieve this, new domain specific knowledge should be added, such as new shared literals, new cost computations and/or new virtual actions, but their usage would be similar.

We believe that this framework will provide latitude for even more improvements and for devising more elaborate techniques to reduce computation time and focus on the more promising alternatives. For instance, a way to extend our work would be to choose a backtracking point using information from both the geometric and symbolic contexts. As of now, the maximum allowed number of alternatives is fixed by the domain expert, while it could be computed or learned from the geometric and symbolic contexts.

REFERENCES

[1] R. Alami, "On human models for collaborative robots," in *CTS Int. Conf. on Collaboration Technologies and Systems*, 2013, pp. 191–194.

[2] M. Warnier, J. Guitton, S. Lemaignan, and R. Alami, "When the robot puts itself in your shoes. managing and exploiting human and robot beliefs," in *IEEE Int. Symp. on Robot and Human Interactive Communication*, 2012, pp. 948–954.

[3] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, "Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction," *Int. Journal of Social Robotics*, 2011.

[4] E. A. Sisbot and R. Alami, "A human-aware manipulation planner," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1045–1057, 2012.

[5] L. de Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining HTN planning and geometric task planning," *CoRR*, vol. abs/1307.1482, 2013.

[6] L. De Silva, M. Gharbi, A. K. Pandey, and R. Alami, "A new approach to combined symbolic-geometric backtracking in the context of human–robot interaction," in *IEEE int. conf. on robotics and automation*, 2014.

[7] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *IEEE int. conf. on robotics and automation*, 2010, pp. 5002–5008.

[8] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, "Smt-based synthesis of integrated task and motion plans from plan outlines," in *IEEE int. conf. on robotics and automation*, 2014.

[9] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Int. Workshop on the Algorithmic Foundations of Robotics*, 2014.

[10] C. Dornhege, A. Hertle, and B. Nebel, "Lazy evaluation and subsumption caching for search-based integrated task and motion planning," in *IROS workshop on AI-based robotics*, 2013.

[11] C. Dornhege, P. Eyerich, T. Keller, M. Brenner, and B. Nebel, "Integrating task and motion planning using semantic attachments." in *Bridging the Gap Between Task and Motion Planning*, 2010.

[12] 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 *ICAPS workshop, Combining Task and Motion Planning for Real-World Applications*. Citeseer, 2012, pp. 13–20.

[13] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE int. conf. on robotics and automation*, 2011, pp. 1470–1477.

[14] 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 *IEEE int. conf. on robotics and automation*, 2011.

[15] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," *IEEE int. conf. on robotics and automation*, 2014.

[16] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *IEEE/RSJ int. conf. on Robots and Systems*, 2012, pp. 957–964.

[17] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," *IEEE/RSJ int. conf. on Robots and Systems*, 2014.

[18] E. A. Sisbot, R. Ros, and R. Alami, "Situation assessment for human-robot interactive object manipulation," in *IEEE Int. Symp. on Robot and Human Interactive Communication*, 2011, pp. 15–20.

[19] M. Daoutis, S. Coradeschi, and A. Loutfi, "Cooperative knowledge based perceptual anchoring," *Int. Journal on AI Tools*, vol. 21, 2012.

[20] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *ECAI*, vol. 16, 2004, p. 895.

[21] F. Gravot, S. Cambon, and R. Alami, "asymov: a planner that deals with intricate symbolic and geometric problems," in *Robotics Research*. Springer, 2005, pp. 100–110.

[22] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[23] D. Nau, M. Ghallab, and P. Traverso, *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann, 2004.

[24] R. Lallement, L. de Silva, and R. Alami, "Hatp: An htn planner for robotics," in *ICAPS Workshop on Planning and Robotics*, 2014.

[25] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 874–883, 2007.