Extending CHIMP (Conflict-driven Hierarchical Meta-CSP Planner) with Spatial Solver

Hatem Htira

University of Bamberg, Kapuzinerstraße 16, 96047 Bamberg, Germany

Abstract. The robots' environment contains a different type of knowledge that we need to consider in planning our autonomous mobile robot. In this context, we consider a CHIMP planner that combines hybrid reasoning, which fuses symbolic, temporal, resource planning, and HTN planning to reduce the search space. We extend the CHIMP domain towards spatial reasoning to achieve tasks within the proposed architecture merge between the robot point and the object point of view. Therefore, we include geometric information about the position and the size of the objects to the meta-CSP knowledge (causal, temporal and resource knowledge, and knowledge provided by an external path planner). This paper will tackle the absence of geometric reasoning by integrating this solver into the hybrid HTN planner.

Keywords: Hybrid planner · HTN · Geometric planner · Rectangle Algebra.

Table of Contents

1 Introduction

Raise environmental awareness of the different varieties of knowledge for the autonomous robot in the real world is essential to solve complex tasks. In this sense, we base our in this paper on hybrid hierarchical planner CHIMP introduced by Sebastian Stock [6] based on the hybrid planner [5] of Mansouri and Pecora, and HTN decomposition to provide other solution instead of using purely causal planning as in the RACE project. This general approach to such a hybrid planner, which is implemented as further meta-CSP, involves a huge search space that will be addressed and solved through the decomposition technique introduced by HTN to minimize this hybrid search space. This framework empowers the system to be efficiently and reliably implemented but comes at the expense of low flexibility to respond to the demands of dynamical environments such as human-populated or partially-known environments. Moreover, the CHIMP planner suffers from the absence of spatial reasoning.

A motivating example: a robot must put some objects on a table. Without geometric reasoning on every element's position and size, it has no way to know if the table will be big enough to cover all the objects. The existing solution presents a naive approach to deal with this limitation by discretizing the geometric world state and work only at the symbolic level with the (finite) set of discrete positions such literal isOn(Table). Our solution here is to integrate a Spatial solver into our Hybrid planner to test every method and operator's geometric feasibility through spatial fluent and constraints to determine the prior action by the backtracking algorithm.

2 Hierarchical Task Network (HTN)

In various fields such as robotics and mission control, hierarchical Task Network (HTN) planning is a popular approach that specifies high-level guidance on how robots and agents should execute tasks while also giving the planner enough adaptability to choose the lower-level actions and their ordering. In many challenging domains, the requirement for rich domain knowledge to characterize the world allows HTN planning to be very beneficial and also to perform presents much better complexity scaling in comparison with classical planning. The idea here is to represent a plan as a partial ordering of actions with respect to time, instead of the classical view of a plan as a linear sequence of actions that are executed one at a time.

The planning process [2] originates by decomposing the goal task by reducing it into several sub-task networks until all compound tasks are decomposed, and hence, a solution is attained. A set of non-decomposable primitive (non-primitive) tasks are added to the initial world state to construct the solution.

HTN planning includes an initial state or problem description, an initial task network that has to be decomposed and represents the goal to achieve, and a domain knowledge consisting of how to decompose the task networks. The HTN domain description contains three kinds of knowledge artifacts: axioms, operators, and methods. The axioms are similar to logical Horn-clause statements:

the planner uses them to infer conditions about the current state. The operators are like the planning operators used in any classical planner. The names of these operators are designated as primitive tasks. The methods rely on a description of the problem based on a set of tasks using a hierarchy that resembles the way humans reason about problems by decomposing them into sub-problems depending on the context. The planner can decide whether a method is applicable by simply checking that the current state meets all its preconditions. This allows early pruning of search space(compared with the classical approach) without impacting its completeness. Moreover, if preconditions are met, it then branches from the abstract level to the subsets of smaller tasks(which may be either primitive or non-primitive tasks) to accomplish, along with some constraints over those tasks that must be satisfied as their ordering (Ordering s0 s1) or to specify the argument options as (Values ?object knife1 knife2) or other hybrid constraints.

Indeed, in this hierarchical settings, a significant benefit is the ability to express constraints between distinct actions by placing them in the same higher-level action, e.g., a *get_object* action is always followed by a *put_object* action in all task networks it appears in. Most of the HTN planning research has been focused on practical application and computer games due to real-time or fast response requirements. Some of the best-known HTN-planning systems are The SHPE (Simple Hierarchical Planning Engine) planner, O-Plan2, and SHOP2.

However, HTN planning is more expressive than STRIPS-like planning in theory and is based purely on causal knowledge. However, none of the hierarchical planners can achieve such expressiveness in practice. As an example, resource and temporal constraints are supported by few planners up to a certain level, temporal reasoning is supported by O-Plan2 and also explicitly in SHOP2.

One of the limitations of this symbolic planning, When computing human-aware robot plans is the inability to reason about the actual impact of the action on the environment and allow to take into account the different types of knowledge available such as geometric, temporal, and resource feasibility in the current world state, which raises the necessity to tackle this limitation through extending the hierarchical plan with hybrid reasoning.

3 Hierarchical Hybrid Planner CHIMP

Due to the requirements to extend the HTN planner with hybrid reasoning, we introduce a constraint satisfaction problem (CSP). This CSP is called meta-CSP that represents a hybrid problem in various levels of abstraction. The planning problem is defined in a constraint network and tries with ordinary constraint solvers to find a plan as a valid solution for the constraint network. This Meta-CSP approach for hybrid reasoning in several constraint networks was taken up, generalized, and formalized by Mansouri and Pecora at the University of Örebro. This approach has the leverage to enforce other kinds of high-level requirements by adding further meta-constraints. Due to this reasoning, we will address the different forms of knowledge in an integrated way instead of handling them separately. Nevertheless, that approach lacked advanced causal planning capabilities and considering all of these other forms of knowledge leads to a vast search space. As a motivation example, when we have a restaurant service robot, it must interfere that the hot coffee should be delivered before it is cooldown or the guest gets impatient: perform causal and temporal reasoning jointly, and that it has to use a tray when it has to carry more than two dishes: sense about resources jointly with time.

This meta-CSP framework has recently been used as a base to create the hierarchical hybrid planner CHIMP (Conflict-driven Hierarchical Meta-CSP Planner) [6]: a (pure) HTN planner with a timeline-based representation which is based on meta-CSP planning to represent the hybrid plan space and uses hierarchical planning as the strategy for cutting efficiently through this space.

The bird's eye view of CHIMP is to combine the advantage of these two approaches and overcome the extensive hybrid search by extending it with a hierarchical task decomposition strategy as a powerful means to guide their exploration of the search space. It joins hierarchical causal knowledge, temporal knowledge, resource, and state variable scheduling, as well as data from external sources, such as the average time it would take the robot to drive between two poses. For this, CHIMP uses a shared constraint network to merge symbolic and temporal constraints. The HTN decomposition technique, scheduling, and calls to external reasoners are applied as meta-constraints. A CSP-style backtracking search is applied to get a plan in which the common constraint network is temporally and symbolically consistent and also fulfils all the requirements represented by the meta-constraints.

The temporal requirements can explicitly be formed with relations from Allen's interval algebra with metric bounds composed of 13 atomic relations to describe the temporal relationship between two intervals(fluents). This results in a disjunction of several relations which will be suitable to represent the timeline constraints between preconditions and effects fluents. Also, we have causal constraints that present another form of representation used by HTN-planner to specify the hierarchical causal domain knowledge, which consists of additional symbolic and temporal constraints as dc, pre, opens, closes, ordering matches. Otherwise, we determine the resources in domain definition by specifying the name of the resource n and its holding capacity k (Resource n k)as an example of the maximum holding capacity if the leftArm of a robot.

In CHIMP's expressive domain description language, the robot's actions and requirements can be modelled in a fine-grained manner and, for example, the exact qualitative and quantitative temporal relationships mapping between tasks, preconditions, and effects.

4 Extension of CHIMP with Spatial reasoning

4.1 Motivation and Proposed Solution

The current version can reason only about causal, temporal and resource knowledge and use the knowledge provided by external reasoners. As it is based on the more general meta-CSP reasoning approach, it can even be extended with other forms of knowledge, and in this work, we integrate spatial reasoning into the CHIMP project. The inability to reason about the real world's geometry to test the plan's geometric feasibility. As a motivation example: a robot must put some objects on a table. Without geometric reasoning, the robot has no way to know if the table will be big enough for all the objects or to reason about which Poses close enough to an object to grasp it or determine the shortest perfect path avoiding a collision. It is essential to point out that many different configurations can be interpreted from a single symbolic predicate, for instance, *On(?object ?plArea)* any position of an object ?object on top of a given placing area, it can have any x and y in the bounds of the table surface.

To address this problem, we provide geometric representation to the symbolic state by adding extra spatial constraints to the meta-CSP knowledge as the assignment of an object's position before and after moving it. The solution here is to determine a plan that fulfils all the causal, temporal and spatial constraints and provide a consistent constraint network.

We propose a constraint-based approach for the two-dimensional ARA+ (Augmented Rectangle Algebra plus) [5] also introduced by M. Mansouri and F. Pecora. This approach presents 196 relations between two bounding boxes whose sides are parallel to the axes of some orthogonal basis in a 2-dimensional Euclidean space, which grants an extension of Allen's Interval Algebra to represent spatial information in the form of a spatial constraint network. This spatial formalism calculates the IA relations of the projection of two 2D-rectangles to their XY-axes segments. These spatial calculi combine qualitative and metric knowledge to obtain solution expressed in actionable metric terms.

Among predicates with spatial semantics, we recognize the predicate "on" and distinguish it spatial fluent with metric bounding boxed, which reflects the spatial knowledge on the entities' placement. For illustration, in Figure 6, two rational rectangles satisfy the fundamental relation (Meets,Before).

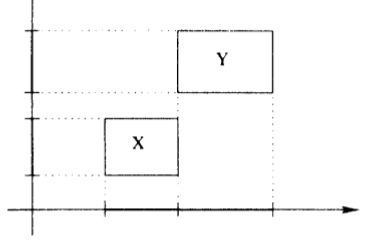

Fig. 1. Illustration of RA.

The central problem is to know whether the spatial constraint network is consistent or, i.e., whether or not the spatial data represented by the network is coherent. If we back to the CHIMP context, The planner will be able to reason whether a method/operator is applicable by simple checking that all its preconditions, including the spatial fluent, are met by the current state and then apply the geometric effect of executing a specific task: as pickingUp(mug): the previous position of the mug is no more valid. Moreover, ARA+ includes besides binary relations also unary relations, which defines the size and the absolute placement of objects.

In the next section, we will explain integrating the proposed spatial planner with CHIMP's meta-CSP and how it has expressed in both domain and problem representation language.

4.2 Extended Problem Representation

A CHIMP planning problem [6] is a triple $P = ((F, C), T, D)$. It is (F, C) a constraint network consisting of fluents F, which describe the initial state and the tasks, and causal, temporal and symbolic constraints C. Furthermore, $T \subseteq F$ is a set of task fluents to be fulfilled, and D is a CHIMP planning domain.

Here we added extra spatial fluent [4] to describe the spatial knowledge of the objects in the scenario. We start first with unary constraints with present the placement and the size of the objects: the unary relation

$$Size[L_x, U_x], [L_y, U_y]$$

bounds the distance within two points of the same rectangle along one axis, constraining the lower and upper bound on both XY-axes. in our application, we assume that $L_x = U_x$(width) and $L_y = U_y$(length). The unary relation

$$At[L_{x1}, U_{x1}], [L_{x2}, U_{x2}], [L_{y1}, U_{y1}], [L_{y2}, U_{y2}]$$

bounds the absolute placement of spatial entities.the first and second bounds represent successively the interval of projection of the two points along X-axe , and analogously the the third and fourth interval on y-axis

```
1 (UnarySpatialFluent sp1 Size(knife1 [4,4] [18,18]))
2 (UnarySpatialFluent sp2 Size(fork1 [4,4] [18,18]))
3 (UnarySpatialFluent sp3 At(knife1 [50,50] [54,54] [50,50]
      [68,68]))
4 (UnarySpatialFluent sp4 At(fork1 [72,72] [76,76] [50,50]
      [68,68]))
```

Listing 1.1. Unary relations

An excerpt of unary relations in problem description is given in Listing 1.1. The first two lines indicate the size of *knife1* and *dish1* in the scenario and the line 3 and 4 contain fluents stating the absolute placement of our two objects. when we focus on knife1 has (line 1) the length 18 and width 4 and it is placed on position in which the start x_1 and end x_2 points of the projection on X-axe (line 3) should be equal or larger to the width w of the knife1 $= x_1 - x_2 \leq w$, in this case $(50 - 54 \leq 4)$, and also analogous to the y-axe with y_1 and y_2. It makes no sense to have the length of the interval is null. Also, it is possible to have the reserved placement of the object is larger the object, we could here theoretically take consideration of a margin of error of placement of an object as reserving larger placement, and the robot has to place the object inside this interval without specifying the exact position.

It is important to notice that the object's size is fixed here, but in comparison with the position, it could be updated based on the task.

```
1 (RectangularSpatialFluent sp5_x Before(knife dish))
2 (RectangularSpatialFluent sp5_y During(knife dish))
3 (RectangularSpatialFluent sp6_x After(fork dish))
4 (RectangularSpatialFluent sp6_y During(fork dish))
5 (RectangularSpatialFluent ass1_x Equals(knife knife1))
6 (RectangularSpatialFluent ass1_y Equals(knife knife1))
```

Listing 1.2. Binary relations

Listing 1.2 shows an example of using binary spatial constraints, spatial relation between two objects is parsed in two lines in which every line describe the projection axe. Each relation is a disjunction of Allen basic relations modelling the knife and dish's mutual placement (as in line 1 and 2). It must be pointed we can not model negation of spatial relations as "knife should not be on the right side of the dish".

We also use Description logics (DL) to design and implement our model to represent our spatial knowledge: in the parsing process, each spatial variable/object will be created, and then we assign the unary relations to each variable, and then we specify the binary relations between different objects. when we focus about the the last two lines of this Listing, we after creating our two variables *knife(T-box)* and *knife1(A-box)*, we assign the constraints(*Size* and *At*) to knife1 as defined in Listing 1.1 and then we specify that knife1 is instance of knife with *(Equals,Equals)* binary relation.

We applied DL in problem description because it is expressive to model flow of information among different spatial contexts: we could have different tables in which every table has its own settings(Formal, Buffet, Breakfast..) and also we have a counter setting, in our example, we specified that the knife is on the right side of the dish and the fork on the right side (the binary relation between our T-box objects) as we have in first four lines in the second listing and these objects exist on *placingArea1*. Furthermore, when executing the task *put_object(knife1 placingArea1)*, the planner has to place the knife1 respecting the designed layout, and here we assign it to the knife. And when we want to execute the task *get_object(knife1)* in this context, the robot will hold the knife1 and the assignment between the knife1 and knife will be deleted, and the table setting will be always intact.

4.3 Extended Domain Representation

A CHIMP planning domain is a 4-tuple $D = (O, M, RC, SV)$, which consists of a set of operators O, a set of methods M, a set of resources RC and a set of state variables SV. The domain is assumed to be static and specific to the robot's capabilities and environment, aiming to solve the given planning problem. Moreover, here we extend the usual preconditions and effects to a rich spatial setting.

The resulting representation in the upcoming extraction Listing 1.3 will be able to reason about the spatial network constraints. We demonstrate the operator's spatial constraints for tasks with the name *!pick_up_two_object*. We have two objects on top of each other, and the robot has to hold each object in a different hand. After checking collision between the two objects (it will be explained in the next subsection), the robot must hold the upper object first then the second. The keyword $DelSC$ precise the spatial binary and unary constraints as negative effects which by applying the operator their constraints will be deleted. The spatial binary negative effect is defined as

$$(DelSC(At)(?objName))$$

in which we will delete the existing object's position (line 8 and 9) because it is not valid anymore after holding the specified object. For binary negative constraints (line 7), we delete all the binary constraints between the two objects: obj1 on top of obj2 $(During, During)$ and also in the parsing process, we delete the assertion of each object $(Equals, Equals)$, e.g., $Knife1 \rightarrow (Equals, Equals) \rightarrow knife$. The existing constraints to delete should be already be defined in the problem definition.

```
 1  (:operator 10
 2   (Head !pick\_up\_two_object(?obj1 ?arm ?obj2 ?otherArm))
 3   (Pre p1 On(?obj2 ?fromArea))
 4   (Pre p2 RobotAt(?mArea))
 5   (Pre p3 Connected(?fromArea ?mArea ?preArea))
 6  .....
 7   (DelSC (?obj1,?obj2))
 8   (DelSC (At) (?obj1))
 9   (DelSC (At) (?obj2))
10  )
```

Listing 1.3. Example of an operator for picking up two objects.

An excerpt of adding binary spatial constraint is given in the Listing 1.4, when we execute the task of placing an object as placing the mug on specified placing area, the planner needs to assert the object the existing associated Terminology Box definition (mug in this case). Moreover, we affect this assertion in line 6 through the Rectangle Relation *(Equals,Equals)*. Additionally, the robot must be holding mug1 with the given arm. This is declared with the keyword *Values* that restricts the domain of the symbolic variable *?obj* to the symbol mug1 or mug2. Also, we need to define a specific placing object operator for each kind of T-box.

```
(Head !place_object(?obj ?arm ?plArea))
 (Pre p1 Holding(?arm ?obj))
 (Values ?obj mug mug1 mug2) # holding only mugs
 (Pre p2 RobotAt(?mArea))
...
 (SC (Equals,Equals) (?obj,?Mug)) #add Assertion mug1—>Mug
 (Values ?Mug Mug)
)
```

Listing 1.4. Example of an operator for placing an object.

In this example of the operator *!place_object(?obj ?arm ?plArea)*, we considered the case when we are placing a dish between fork and knife and the size of the dish is larger than the distance between these two objects. Here we need to shift one of the other objects, and in order to achieve it, we defined a new operator called *!shift_object* with added extra spatial effect *(SC ?objFork = ?objDish + C)* to move the placement of the fork and then we place the dish. With this spatial constraint, we shift the current position of the fork to the left side from X-perspective based on the dish size: the defined position of the fork (x_1, x_2) (through the unary constraint At) will be incremented with the width of dish w $\pm$ constant C.

4.4 Spatial Fluent

To ensure more expressiveness, we extend the problem representation with additional spatial fluents to check complex conditions such check collision check enough space between two objects, reachability, visibility. In this project, we provide a possible solution for some restaurant service robot scenario. To achieve the task to put a dish between a knife and fork, we need to take into consideration the case when that no enough distance for it. The solution here is to add a spatial function that will be evaluated during the parsing process. In this case, we expressed this fluent as

(Fluent f8 enoughDistance(dish1 knife1 fork1 undef))

Here, we will check if the size of the first argument fit between the second and third argument. We start in problem with *undef* which will be evaluated and replace it with true or false. This fluent will be checked in preconditions of the method *put_object* as indicated in the Listing 1.5. if it is evaluated to false and the distance is not enough, we will need to shift the third argument (fork1) based on the dish's size with an empty arm as explained in the previous subsection. This task is modelled by decomposing the goal task into several ordered subtask as indicated in this extraction of this method.

```
(:method
  (Head put_object(?objDish ?plArea))
  (Pre p1 Holding(?arm ?objDish))
  (Pre p4 enoughDistance(?objDish ?objKnife ?objFork ?false))
  (Values ?false false)
  ....
  (Sub s1 drive_robot(?preArea))
  (Sub s2 assume_manipulation_pose(?manArea))
  (Sub s3 !shift_object(?objFork ?objDish ?armEmpty))
  (Sub s4 !place_object(?objDish ?arm ?plArea))
```

Listing 1.5. Example of using a spatial function to check enough distance.

With the same logic, we introduce the next scenario to serve an empty mug as specified in Listing 1.6. The idea here is to consider when we have a knife inside the mug, which we need to emplace it on the table and hold only the mug. To achieve that, we added spatial function to check collision

(Fluent f8 isCollision(knife2 mug1 undef))

To determine here if an object on top/inside of the other in this bidimensional Rectangle Algebra, we need to check if we have either of these two relations *(During,During)* or *(Equals,Equals)*. Moreover, as the previous function, it will be evaluated, and if we have knife2 inside mug1, then we need to pick up first the uppermost object (or the inside one) and then the second one(line 6). Subsequently, we place the knife2 on the same table(line 7), and then we serve the empty mug to the target area *?toArea*.

```
 1  (:method
 2    (Head serve_an_empty_mug(?obj2 ?toArea))
 3   (Pre p6 isCollision(?obj1 ?obj2 ?true))
 4   (Values ?true true)
 5    ....
 6   (Sub s1 !pick_up_two_object(?obj1 ?arm ?obj2 ?otherArm))
 7   (Sub s2 put_object(?obj1 ?fromArea))
 8   (Sub s3 drive_robot(?preArea1))
 9   (Sub s4 assume_manipulation_pose(?mArea1))
10   (Sub s5 !place_object(?obj2 ?otherArm ?toArea))
```

Listing 1.6. Example of using a spatial function to check collision.

5 Evaluation

In order to evaluate the effectiveness of this approach, we consider the same restaurant domain scenario in which two arms robot is characterised with capacities Cap(LeftArm)=1, Cap(RightArm)=1 and methods of picking and placing objects are modelled.

In our experiment, the robot was given the task to serve an empty dish from counter1 to table1 based on the geometric conditions. The counter is located away from the table in the Counter1, we two objects mug and dish. The geometric planner needs to verify whether the dish is empty or the mug at it through the spatial fluent isCollision as described in the previous subsection. Then the robot holds the dish and moves to table1. In this step, we have two objects (fork, knife) and the held dish. At this point, the goal of the robot is to place the dish between the fork and the knife if the distance between the two objects is enough by the spatial Fluent isEnough. The robot knows all the named Objects' positions and can move between different locations freely in this Obstacle-free environment.

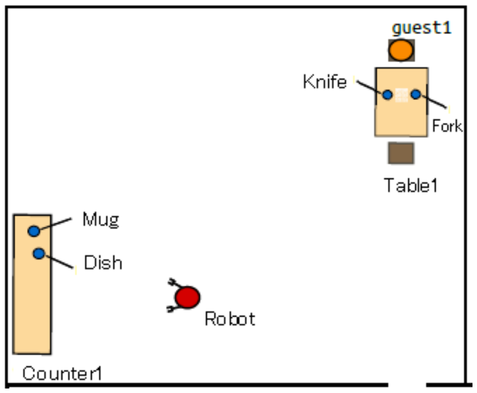

Fig. 2. Illustration of the scenario.

Figure 2 shows the floor layout of the environment and the initial situation, which lead us to 4 possible scenarios based on the evaluation of two Boolean functions isCollision and isEnough. Scenario A is modelled so that the mug is near the dish on the counter1 and that distance between the knife and fork is far away, which opposite to the D scenario. B is composed of the mug on the dish, and the distance is enough and C designed that the dish is empty and the distance is not enough.

In C and D problems, the planner is required to shift the knife to the left according to the new placement determined based on the dish size. In B and D, the robot must empty the dish by holding the mug and the dish in a separate arm and then place back the mug on the counter. As a consequence, the length generated plan of B is larger than A with an extra operator put_down(mug1 counter), and D larger than A with two operators put_down and shift(knife) as we can observe it in Figure 3.

The resulting statistical computation times for the complete experiment are listed in Figure 3 and 4 which indicates the combined planning time. The serving action consists of the planning times for reaching for the dish, lifting it, and planning to the target positions and checking enough distance to place it for

		Numbre of Spatial Fluents (Spatial Constraints + Fluents)					
Scenario	Plan length	29	33	37	41	48	50
A	10	317	340	380	401	387	402
B	11	350	380	400	390	407	424
C	11	474	504	495	537	528	571
D	12	498	552	590	595	662	640

Fig. 3. Performance results of the spatial solver.

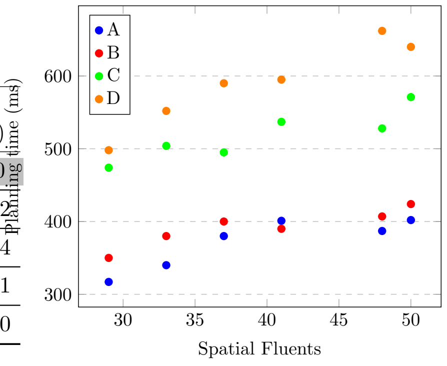

Fig. 4. Performance of the planner.

each scenario. The idea here in each run we increment the number of spatial objects and the number of the spatial constraints between the objects (binary and unary constraints) with fixed plan length. We start in the first run with the minimum possible constraint to execute the spatial functions, and the planner will be able to empty the dish successfully and shift the knife when needed.

Generally, the measurement time increases with the number of the used operator of generated solution [6]. We have measured the meantime for plan generation for the four input problems. The table in Figure 3 shows times dependent on incrementing spatial fluent number in each iteration with fixed solution length. It can be seen that as less the number of spatial fluent is, the faster it is for the planner to generate a solution.

As more spatial constraint we have, the geometric planner requires more time to check the consistency between different spatial constraints. All the four represented problems are geometrically consistent. The planner is implemented in the way to check the geometric consistency in each chosen operator: the spatial solver will be able to reject described problem at t0 when the geometric consistency is not guaranteed, and also in case of shifting the knife, for example, if the knife hangs already on the edge of the table. If an inconsistent state is determined, then no solution will be generated. Due to the hybrid reasoning approach, if a solution is generated, it will be temporal, symbolic, resource and spatial feasible.

Listing 1.7 shows the required 11 operators of the plan with their predicates and flexible temporal intervals of problem C. As we have no collision between the mug and dish, we pick up only the dish1 (line 1). Thanks to its partial-order planning capabilities, CHIMP was able to obtain "operator parallelism", which leading to a shorter execution time of the overall plan as we have it with the two operators !move_torso and !move_arm_to_side (line 6 and 7). Line 10 indicates the required operator to shift fork1 to the left side to enlarge the distance to have enough placement for dish1.

```
!pick_up_object(dish1 rightArm1) [[1,1],[4001,271988]]
!move_base_blind(preManipulationAreaEastCounter1)
    [[4002,271989],[8002,275989]]
!move_torso(TorsoMiddlePosture) [[8003,275990],[12003,279992]]
!move_arms_to_carryposture(n) [[8005,275992],[12005,279992]]
!move_base(preManipulationAreaSouthTable1)
    [[12006,279993],[15006,282993]]
!move_torso(TorsoUpPosture) [[15007,282994],[19007,290995]]
!move_arm_to_side(leftArm1) [[15007,282994],[19007,286994]]
!move_arm_to_side(rightArm1) [[19008,286995],[23008, 290995]]
!move_base_blind(manipulationAreaTable1)
    [[23009,290996],[27009,294996]]
!shift_object(fork1 dish1 rightArm1)
    [[27010,294997],[32010,297999]]
!place_object(dish1 rightArm1 placingAreaTable1)
    [[32011,34010],[37012,299999]]
```

Listing 1.7. Required plan's operators to execute task C.

The Figure 6 indicates the first hierarchical level of the given solution in listing 1.7, the task decomposes into subtasks of ordered methods(yellow boxes) and operators(blue boxes). This decomposition guides the search process and reduces the hybrid search space. This design of this hierarchy is modelled generally by the designer, and it could be updated. In addition to this, its expressivity helps understand the task and the hierarchy decomposition even by people outside the planning area. In our HTN planner, the search depends on the order and the operators' duration, determined by the temporal reasoning, on which the search nodes are expanded.

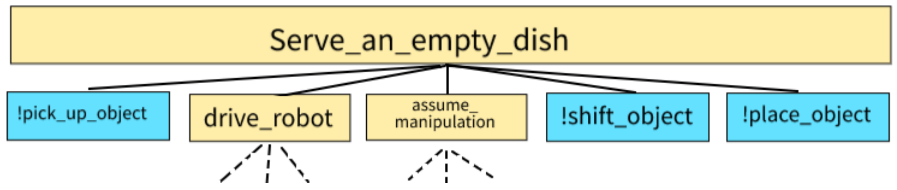

Fig. 5. Example of first HTN decomposition level for serving an empty dish.

6 Improving The Spatial Solver Complexity

In regards to the computational hardness of the full Allen algebra and solve the problem in polynomial time. We propose other solution to define two-dimensional rectangular objects. We present Disjunctive Linear Relations (DLRs) to reason about spatial constraints. Allen relations used in Rectangle algebra, presented in chapter 4, can be easily expressed by disjunctions of the form: $x_1 \leq x_2 \vee y_1 \leq y_2$. The idea here is that the geometry layout and the object's position and size will be defined through algebraic equation and inequality.

As by RA, DLR objects are rectangles parallel to XY-axes in which four variables define every object. The domain of the variables are bounded intervals so that the lower bound of the interval is less or equal to its upper bound $Lx \leq v \leq Ux$, the bounds here are integers. The solution can be found by consistency methods to determine variables that satisfy all the inequalities. The spatial relations are expressed by Boolean combinations of the algebraic constraints between the rectangle's x and y intervals, which presents (analogically as Allen Interval Algebra) 13 relations between two rectangles interval.

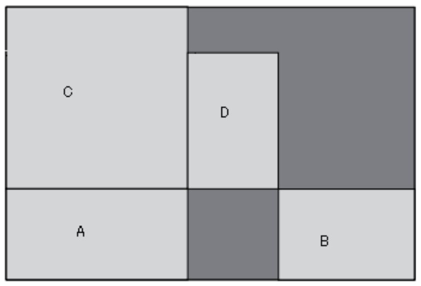

Fig. 6. Example of topological relation between objects.

Figure 6 shows the example packing of rectangles which we will define this geometric scenario in both RA and DLR to compare the two problem representations. In linear programming, we ensure the rectangle (A) size with its unary and binary Constraints is described only by two sets of inequalities in line 2 and 9. and we can also notice that the "after" relation for X axe (line 17) is expressed through determining the distance between the edges of the two rectangles (line 3) and Equals (line 18) through the equality in line 7 and 8. For both methods, we can determine the size of the two rectangles' global container and determine its positions in comparison to the bounded space XY-axes as "dish on the table".

```
//size of the rectangles A and B
Bounds A_size_x = new Bounds(20,20);
Bounds A_size_y = new Bounds(10,10);
Bounds B_size_x = new Bounds(15,15);
Bounds B_size_y = new Bounds(10,10);
UnaryRectangleConstraint sizeA = new
     UnaryRectangleConstraint(
     UnaryRectangleConstraint.Type.Size
     ,A_size_x,A_size_y);
sizeA.setFrom(A);
sizeA.setTo(A);
allConstraints.add(sizeA);

UnaryRectangleConstraint sizeB = new
     UnaryRectangleConstraint(
     UnaryRectangleConstraint.Type.Size
     ,B_size_x,B_size_y);
sizeB.setFrom(B);
sizeB.setTo(B);
allConstraints.add(sizeB);
//Position of A in relation to B
RectangleConstraint AtoB = new
     RectangleConstraint(
         new AllenIntervalConstraint(
              AllenIntervalConstraint.
              Type.Before, new Bounds
              (10, 10)),
         new AllenIntervalConstraint(
              AllenIntervalConstraint.
              Type.Equals));
AtoB.setFrom(A);
AtoB.setTo(B);
allConstraints.add(AtoB);
```

```
//For the x perspective
Ax1 + 20 <= Ax2
Ax2 + 10 <= Bx1
Bx1 + 15 <= Bx2

//For the Y perspective
Ay1=By1
Ay2=by2
Ay1 + 10 <= Ay2
```

Thus, we explore in this section a simple example of horn Disjunctive Linear Relation [3, 1]: TCSPs (Temporal Constraint Satisfaction Problems) is a particular class of CSP where variables represent times and constraints represent relations between two-time points. To elaborate, the unary constraints will be binarised, making the constraints of a TCSP exclusively binary.

To solve the TCSP problem, we need to define a set of assignment, each of which assigns a time point to an event, so all constraints are satisfied. In Spatial Context, the basic idea is to formulate the problem in Euclidean space. Consequently, the constraints and the solutions to TCSP become geometric objects in the space.

The measurement time increases with the number of constraints. We have measured the meantime for different problems. We notice here that TCSP take more time to generate a solution than RA, and the reason behind it is that to order to reason about a two-dimensional rectangle. We used two solvers, one for each axe.

It is important to keep in mind that the method builds on solving linear programs such as TCSP can be solved in polynomial time. However, it is a widespread belief [3] that such calculations are computationally heavy.

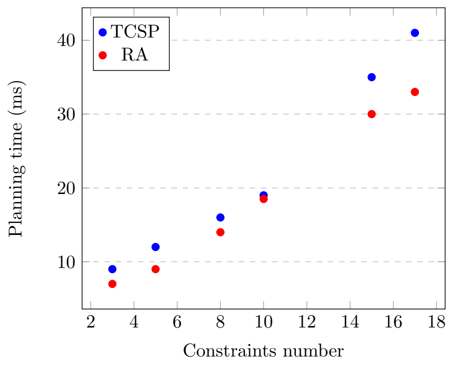

Fig. 7. Performance of TCSP vs Rectangle Algebra.

7 Conclusion and Outlook

We have presented an extension of the CHIMP hybrid planner through providing an additional ground CSP for spatial reasoning, allowing them to reason on the relative placement and geometry of objects for service robot tasks. The Rectangle algebra was in chapter 4 introduced to reason Two Dimensional rectangles. This model provides rich and flexible semantics for encoding the spatial aspect of planning using unary and binary constraints combined with spatial fluents function to define an additional expressive form of knowledge. Notwithstanding this huge hybrid search space, this extended CHIMP's hierarchical planning keeps planning time low. Moreover, it has required more human effort to improve its knowledge base to get better performance.

While the spatial planner provides good performance, we introduced a new technique to decrease the complexity and solve the problem in polynomial time.

Perspectives and Future Work. There are diverse directions for extending this spatial planner.

First, we could extend the existing R.A. easily to a greater dimension than two as 3 Dimensional through considering projection of the sides of a block on the axes. The atomic relation between two 3D objects defines 13^3 relations over the XYZ segments of two rectangles.

Second, our work was based on the assumption that we have a deterministic robot environment in which the planning is performed offline without the stress to learn the model. This assumption does not hold in different scenarios as a populated environment or when the actions are non-deterministic.

To deal with uncertainty and ensure collision-free navigation through an unknown environment with obstacles, we could relax this previous assumption and consider a continuous learning environment using Reinforcement learning and combine it with one-step-ahead predictions from the current state in real-time applying Gaussian Process (GP).

Furthermore, we could adapt CHIMP to be integrated into multi-agent environments, being capable of interacting with external agents. As an example, robot1 on the counter prepares the appropriate drink and then gives it to the other robot who is responsible for serving it to a customer considering non-deterministic action, which might eventually lead to unwanted effects as balance disturbances.

References

1. Martin Berger, M Schröder, and K-H Küfer. A constraint programming approach for the two-dimensional rectangular packing problem with orthogonal orientations. 2008.
2. Ilche Georgievski and Marco Aiello. Htn planning: Overview, comparison, and beyond. *Artificial Intelligence*, 222:124–156, 2015.
3. Peter Jonsson and Christer Bäckström. A unifying approach to temporal constraint reasoning. *Artificial Intelligence*, 102(1):143–155, 1998.
4. Masoumeh Mansouri. *A constraint-based approach for hybrid reasoning in robotics.* PhD thesis, Örebro university, 2016.
5. Masoumeh Mansouri and Federico Pecora. A representation for spatial reasoning in robotic planning. In *IEEE International Conference On Intelligent Robots and Systems (IROS)-Workshop on AI-based Robotics, Tokyo, Japan, 3-7 November, 2013.* IEEE, 2013.
6. Sebastian Stock, Masoumeh Mansouri, Federico Pecora, and Joachim Hertzberg. Hierarchical hybrid planning in a mobile service robot. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 309–315. Springer, 2015.