HTN Planning for Pick-and-Place Manipulation

Hisashi Hayashi[1], Hideki Ogawa[1] and Nobuto Matsuhira[2]
[1]*Corporate R&D Center, Toshiba Corporation, 1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan*
[2]*Department of Engineering Science and Mechanics, Shibaura Institute of Technology, 3-7-5 Toyosu, Koto-ku, Tokyo 135-8548, Japan*
{*hisashi3.hayashi, hide.ogawa*}*@toshiba.co.jp, matsuhir@shibaura-it.ac.jp*

Keywords: HTN Planning, Manipulation, Robot.

Abstract: We introduce new heuristics of HTN (Hierarchical Task Network) planning for mobile robots with two arms/hands that pick and place objects among movable obstacles. Based on our new heuristics, the robot moves obstacles if necessary, picks and places the target objects without collisions. The robot chooses the (right or left) hand to use for each manipulation in order to avoid collisions and reduce the number of obstacle movements. In most of the previous approaches that combine task planning and motion planning, collisions between an arm and obstacles are checked only by the lower-level geometric motion planner. Therefore, the high-level general-purpose task planner often produces a plan that is not executable by the lower-level modules. On the other hand, in our new heuristics, the task planner roughly checks collisions, and produces executable plans.

1 INTRODUCTION

Object manipulation among movable obstacles has been an important research area of robotics. Typically, a service robot equipped with hands/arms moves around, picks and places objects in the environment. When reaching a hand to an object, obstacles might exist in the path of the arm. Therefore, it is sometimes necessary to avoid or remove the obstacles.

Traditionally, the combination (Cambon et al., 2009; Choi and Amir, 2009; Haspalamutgil et al., 2010; Hauser and Latombe, 2009; Kaelbling and Lozano-Perez, 2010; Wolfe et al., 2010) of high-level general-purpose task planning and low-level geometric motion planning has played important roles for service robots of this kind. In order to do the given tasks, the task planner makes a rough plan using symbolic information. Typical actions of the task plan are, for example, “go to table1”, “pick object1”, “go to table2”, and “place object1 on table2”. In order to execute these actions, the motion planner controls robots, considering kinematic constraints of the robot and a detailed 2D/3D map of the environment.

However, there is a problem in combining task planning and motion planning. Even if the high-level task planner makes a plan, it is not always possible to execute the plan when the low-level motion planner cannot find a path or a grasp. This forces the task planner to backtrack and call the motion planner many times. This is a serious problem for service robots working in real time. For example, if the task planner calls the motion planner 100 times, and it takes 1 second for each motion planning, then the total time for motion planning will be 100 seconds, which is unacceptable.

This problem arose because the task planner did not consider kinematic constraints of robots and the obstacles in the environment. For example, consider the blocks world problem, a well-known problem of AI and task planning explained in many widely used AI textbooks such as (Russell and Norvig, 1995). A typical task of the blocks world problem is to rearrange the blocks on the table using a robot arm. In order to pick the block A or place a block on top of the block A, the robot has to remove all the other blocks above the block A. However, neither the path of the robot arm nor the collisions between the robot arm and blocks are considered. In reality, the robot stretches an arm from a side of the table and there are many chances of collisions between the arm and other objects on the table.

This paper presents new task planning heuristics for manipulation among movable obstacles. We consider the use of typical mobile robots with two hands/arms. Unlike the previous approaches, the path of the arm and locations of obstacles are roughly taken into account by the task planner. Based on our

Hayashi H., Ogawa H. and Matsuhira N..
HTN Planning for Pick-and-Place Manipulation.
DOI: 10.5220/0004226303830388
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 383-388
ISBN: 978-989-8565-38-9

new heuristics, the robot moves obstacles if necessary, picks and places the target objects without collisions. It is not the aim of this paper to improve efficiency of motion planning algorithms. We would like to make "executable" task plans.

The remainder of this paper is organized as follows: Section 2 describes the problem we solve. Section 3 shows how to solve the problem based on our new task-decomposition heuristics. Section 4 evaluates the heuristics by means of experiments. Section 5 is the conclusion.

2 PROBLEM DESCRIPTION

We consider a mobile robot equipped with right and left hands/arms. There are shelves inside a room. (See Figure 1.) There are some objects on the shelves.

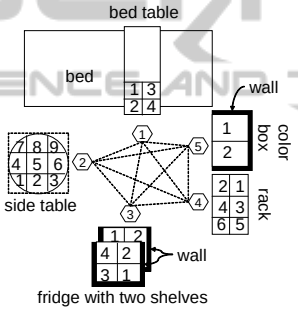

Figure 1: Room.

Each object has an RFID tag and the RFID reader attached under each shelf can recognize rough 2D-grid location of each object on the shelf. Each cell of the 2D grid on a shelf has a unique number. The robot has a sterio camera, and can pick recognized objects. For simplicity, we assume that there does not exist more than one object in each cell of the 2D-grid, and each movable object on the shelf exists inside one cell.

There is a node on the floor in front of each shelf, from which, the robot moves an arm to pick or place an object. Like shelves in a fridge, there might be more than one shelf in front of a node. The robot can freely move from one node to another, localizing itself at each node. Therefore, the problem we solve is mainly for arm manipulation.

We give the robot the task of transferring an object ($object1$) from one shelf ($shelf1$ = side table) to another ($shelf2$ = upper shelf of the fridge). In order to do the task, the robot needs to go to the node ($node2$) of $shelf1$, pick $object1$ with the right or left hand, move to the node ($node3$) of $shelf2$, and place $object1$ on $shelf2$.

The robot cannot grasp the object from above. Therefore, in order to pick or place an object from the side, the robot has to control the arm so that it does not collide with obstacles. We assume two types of shelves. Some shelves have walls like shelves inside a fridge. Some shelves do not have such walls and are like table tops.

We use rough 2D maps to recognize the rough location of each object. However,we do not use detailed 2D/3D-maps for finding obstacles. Instead, we use new task-decomposition heuristics for pick-and-place operations, considering rough geometric and kinematic constraints.

The robot executes "go to", "pick" and "place" using lower-level modules. We do not explain how to implement these actions by lower-level modules including the motion planner. However, we assume the robot can pick and place any object on a shelf with the right (left) hand if there is no obstacle at the right (left) front of the shelf, which will be explained later in Heuristic 2.

3 SOLUTION

This section shows new heuristics for pick-and-place manipulation. We express the heuristics using task decomposition rules of a forward-chaining HTN (Hierarchical Task Network) planning agent called Dynagent[1] (Hayashi et al., 2006). Forward-chaining HTN planners (Hayashi et al., 2006; Nau et al., 1999) make plans by decomposing abstract tasks into more concrete subtasks in the same order that they will be executed. A task decomposition rule expresses how to decompose a task to a sequence of subtasks (= a plan). A task decomposition rule is applicable when its precondition holds.

3.1 Heuristics for Transferring an Object

The robot is given the task of transferring a specified object to a specified shelf. This top-level task will be decomposed as follows where "go to" is an action (= primitive task) and the other subtasks need to be decomposed further by the heuristics that will be defined later. Even if it is possible to pick an object with the right hand, in the later stage of planning, we might find that we cannot place the object with the right hand, in which case, we need to backtrack and consider using the left hand. It might seem that we

[1]Dynagent is a registered trademark of Toshiba Corporation.

have only two alternative plans that use the right or left hand. However, there are many ways to decompose each subtask further. Therefore, there would be more alternative plans.

Heuristic 1. *The abstract task:*

- *Transfer* `object1` *from* `shelf1` *to* `shelf2`.

is decomposed by the following task decomposition rule where `object1` *is an object,* `cell1` *is a cell,* `shelf1` *and* `shelf2` *are shelves:*

- *Task decomposition rule:*
  - *Precondition:*
    - *The robot does not hold an object with the right (left) hand.*
  - *Plan:*
    1. *Primitive: Go to the node of* `shelf1`.
    2. *Abstract: Pick* `object1` *at* `cell1` *on* `shelf1` *without collision with the right (left) hand by the heuristic 3.*
    3. *Primitive: Go to the node of* `shelf2`.
    4. *Abstract: Place* `object1` *on* `shelf2` *with the right (left) hand by the heuristic 5.*

3.2 Heuristics for Finding Obstacles

In Figure 2, the robot is facing one side of a rectangular shelf and picking (placing) the object A from (at) the cell 4. The robot moves the right (or left) arm diagonally from the right (left) front of the shelf. Therefore, if there are other objects or a wall at the right (left) front of the cell 4, there is a high chance of collision. In other words, if there is no obstacle at the right (left) front of a cell, it is not difficult to move the right (left) arm to the cell without collision. This is our key heuristic for finding obstacles.

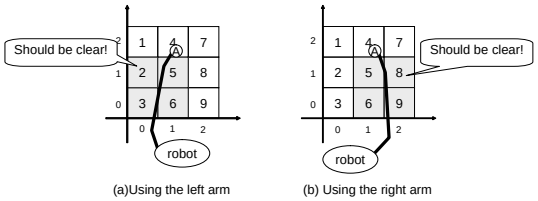

Figure 2: Finding Obstacles.

Heuristic 2. *Suppose that the robot is in front of a rectangular shelf. Let the x-axis of the 2D-grid be the front side of the shelf. Let the y-axis of the 2D-grid be the left side of the shelf. Let k be a non-negative integer. (k is 1 in Figure 2.)*

An object or a wall at the cell (Xr, Yr) *is regarded as a* **possible obstacle for the right arm** *approaching the cell* (Xc, Yc) *if* $Yr < Yc$ *and* $Xc \leq Xr \leq Xc + k$.

An object or a wall on the cell (Xl, Yl) *is regarded as a* **possible obstacle for the left arm** *approaching the cell* (Xc, Yc) *if* $Yl < Yc$ *and* $Xc - k \leq Xl \leq Xc$.

A cell is regarded as **approachable for the right (left) arm** *if there exists no possible obstacle for the right (left) arm.*

3.3 Heuristics for Picking an Object

There are several ways to pick an object. In order to pick an object from a cell on a shelf, the robot has to move an arm to the cell. Even if there is an obstacle at the right (left) front of the cell, if there is no obstacle at the left (right) front of the cell, it is possible to move the left (right) arm to the cell without collisions. Or the robot might be able to remove obstacles.

Heuristic 3. *The abstract task:*

- *Pick* `object1` *at* `cell1` *on* `shelf1` *with the right (left) hand without collision.*

is decomposed by one of the following task decomposition rules where `cell1` *is a cell,* `shelf1` *is a shelf, and* `object1` *is an object:*

- *Task decomposition rule 1:*
  - *Precondition:*
    - *The robot is at the node of* `shelf1`.
    - `cell1` *on* `shelf1` *is approachable for the right (left) arm (by Heuristic 2).*
  - *Plan:*
    1. *Primitive: Pick* `object1` *at* `cell1` *on* `shelf1` *with the right (left) arm.*
- *Task decomposition rule 2:*
  - *Precondition:*
    - *The robot is at the node of* `shelf1`.
    - `cell1` *is not approachable for the right (left) arm (by Heuristic 2).*
  - *Plan:*
    1. *Abstract: Remove all the possible obstacles for the right (left) arm approaching* `cell1` *without collision by the heuristic 4.*
    2. *Primitive: Pick* `object1` *at* `cell1` *on* `shelf1` *with the right (left) arm.*

3.4 Heuristics for Removing Obstacles

This subsection shows the heuristic to remove obstacles. In this heuristic, the robot moves an obstacle to a different cell on the same shelf without collision. The robot can move the obstacle to several different cells.

Heuristic 4. *The abstract task:*

- *Remove* `object1` *at* `cell1` *on* `shelf1` *without collision.*

is decomposed by the following task decomposition rule where `object1` *is an object,* `cell1` *is a cell, and* `shelf1` *is a shelf:*

- *Task decomposition rule:*
  - *Precondition:*
    - *The robot is at the node of* `shelf1`.
  - *Plan:*
    1. *Abstract: Pick* `object1` *at* `cell1` *on* `shelf1` *without collision with the right (left) hand by the heuristic 3.*
    2. *Abstract: Place* `object1` *at another cell on* `shelf1` *with the right (left) hand without collision by the heuristic 5.*

3.5 Heuristics for Placing an Object

In order to place an object on a cell, the arm needs to be able to approach the cell without collision. There is more than one cell to place the object at. (This produces alternative plans.) In addition, we would like to place the object as far as possible from the robot so that the robot can place many objects on the shelf. If the robot places an object near the front side, it might become an obstacle when placing another object. Note that "bottom-left" and "bottom-right" (Baker et al., 1980) are well known strategies for bin-packing.

Heuristic 5. *The abstract task:*

- *Place* `object1` *on* `shelf1` *with the right (left) hand without collision.*

is decomposed by the following task-decomposition rule where `object1` *is an object,* `shelf1` *is a rectangular shelf, the x-axis of the 2D-grid is the front side of* `shelf1`*, and the y-axis of the 2D-grid is the left side of* `shelf1`*:*

- *Task decomposition rule:*
  - *Precondition:*
    - *The robot is at the node of* `shelf1`.
    - *Nothing is at the cell* (X,Y) `shelf1`.
    - *The cell* (X,Y) *on* `shelf1` *is approachable for the right (left) arm (by Heuristic 2).*
    - *The cell* $(X,Y+1)$ *on* `shelf1` *is either off the shelf or not approachable for the right (left) arm (by Heuristic 2) or there is an obstacle at* $(X,Y+1)$.
  - *Plan:*
    1. *Primitive: Place* `object1` *at* (X,Y) *on* `shelf1` *with the right (left) hand.*

3.6 Cost Estimation

Dynagent keeps several alternative plans. When selecting the plan to decompose a task, it selects a plan whose estimated cost is the lowest. This is called best-first search. The cost of a plan is estimated as the sum of the estimated cost of each task in the plan. We can specify the cost of each (abstract or primitive) task. If we do not specify the cost of a task, the cost will be estimated as 0. If we do not provide cost information at all, then it conducts depth-first search.

In manipulation planning, it takes time for the robot to execute an action. Therefore, we would like to reduce the number of action executions. We estimate the cost of each action (= primitive task) as around `c`(>0). Although we estimate the cost of each abstract task as 0, we would like to adjust it more carefully and improve the efficiency in the future.

There are several cells on a shelf. We would like to place the object as far as possible from the front side. Therefore, we slightly reduce the cost when placing an object on the far side of the shelf.

Normally, if there is no wall on the shelf, it is closer for the right (left) arm to approach the right (left)-hand side of the shelf. However, if there are walls on the right- and left-hand side of the shelf, it is easier for the right (left) arm to approach the left (right)-hand side of the shelf without colliding with a wall. Therefore, we adjust the estimated costs of pick-and-place tasks accordingly.

Heuristic 6. *Suppose that the robot is in front of a rectangular shelf. Let the x-axis of the 2D-grid be the front side of the shelf. Let the y-axis of the 2D-grid be the left side of the shelf. Let* `w` *be the width (number of cells) of the shelf. Let* `c`*,* `m` *and* `n` *be positive constants. Suppose that each of* `m`*X,* `m`*(*`w`*-X-1), and* `n`*Y is much smaller than* `c` *where (X,Y) is a cell on the shelf.*

- *The cost of "go to" is* `c`.
- *When there is no wall on the shelf, the cost of picking an object from (X,Y) with the right hand is* `c` - `m`*X.*
- *When there is no wall on the shelf, the cost of picking an object from (X,Y) with the left hand is* `c` - `m`*(*`w`*-X-1).*
- *When there are walls on the right- and left-hand sides of the shelf, the cost of picking an object from (X,Y) with the right hand is* `c` - `m`*(*`w`*-X-1).*
- *When there are walls on the right- and left-hand sides of the shelf, the cost of picking an object from (X,Y) with the left hand is* `c` - `m`*X.*
- *When there is no wall on the shelf, the cost of placing an object at (X,Y) with the right hand is* `c` - `m`*X-*`n`*Y.*

- *When there is no wall on the shelf, the cost of placing an object at (X,Y) with the left hand is* `c - m*(w-X-1)-n*Y`.
- *When there are walls on the right- and left-hand sides of the shelf, the cost of placing an object at (X,Y) with the right hand is* `c - m*(w-X-1)-n*Y`.
- *When there are walls on the right- and left-hand sides of the shelf, the cost of placing an object at (X,Y) with the left hand is* `c - m*X-n*Y`.

3.7 Examples

This subsection shows some plans for pick-and-place manipulation that are made from the heuristics introduced in this section.

Example 1. *As shown in Figure 3, initially, the robot is at the node 1, and the objects A, B, C are at the cells 1, 5, 6 on shelf1, respectively.*

Given the task of transferring A to shelf2, the robot first goes to the node 2, and picks A with the left hand avoiding collision with B and C. Then, the robot goes to the node 3, and places A at 3 on shelf2 with the left hand, following the "far-right" strategy. Note that the robot cannot pick A directly with the right hand because it might collide with B and C.

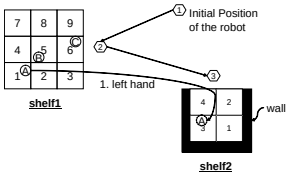

Figure 3: A Plan for Transfering an Object (1).

Example 2. *As shown in Figure 4, initially, the robot is at the node 1, and the objects A, B, C are at the cells 1, 5, 6 on shelf1, respectively. Given the task of transferring B to shelf2, the robot first goes to the node 2. In order to remove the obstacle C, the robot picks C with the right hand, and places C at 7 on shelf1 with the right hand, avoiding collision with B. Then the robot picks B with the right hand, goes to the node 3, and places B at 1 on shelf2 following the "far-left" strategy. Note that the robot needs to use the right hand when placing B at 1 on shelf2 to avoid collision with the wall.*

Example 3. *In Example 2, if initially there is another object D at the cell 1 on shelf2, then the robot cannot place B at 1 on shelf2. In this case, as shown in Figure 5, the robot will place B at 3 on shelf2 following the "far-right" strategy. The robot uses the left hand when picking and placing B. Note that the robot needs to use the left hand when placing B at 3 on shelf2 to avoid collision with the wall.*

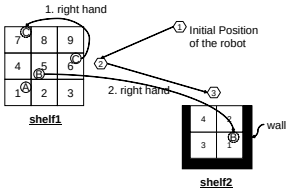

Figure 4: A Plan for Transfering an Object (2).

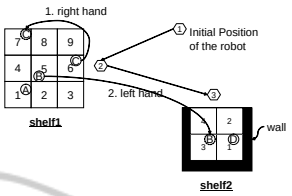

Figure 5: A Plan for Transfering an Object (3).

4 EXPERIMENTS

We tested the scenarios in Examples 1, 2, and 3 using a service robot called SmartPal V[2]. SmartPal V is a mobile robot equipped with two hands/arms. The robot can recognize an object through a stereo camera and pick the recognized object.

We placed SmartPal V in a room shown in Figure 1. The side table and the upper shelf in the fridge in Figure 1 correspond to shelf1 and shelf2 in Figures 3, 4, and 5. The width and depth of each cell on shelf1 are both 150 mm. The width and depth of each cell on shelf2 are respectively 200 mm and 130 mm. The objects (PET bottles) A, B, C and D have RFID tags, and the rough location of each object can be obtained through the RFID reader.

We installed a forward-chaining HTN planning agent, Dynagent (Hayashi et al., 2006), on a PC (Windows XP), and connected it with other modules on SmartPal V through the middleware called OpenRTM-aist 1.0.0.

We confirmed that the robot can execute the plans in Examples 1, 2, and 3 without collision with obstacles. (We also tested other similar scenarios using different shelves in the room.) In Figure 6, the robot is moving an obstacle (C) on a side table so that it can pick the target object (B) without collision with C. In Figure 7, the robot is placing an object (B) on a shelf inside a fridge. In order to avoid the collision with the wall, the robot is using the right hand when placing an object at the far-left corner. Similarly, the robot is using the left hand when placing an object at the far-right corner.

[2]SmartPal is a registered trademark of Yaskawa Electric Corporation.

The planning times in Examples 1, 2, and 3 are respectively 0.594, 0.750, and 0.781 seconds when using a PC equipped with Core2 CPU 2.13GHz and 2.99GB RAM. These planning times are much faster, comparing with the planning times of previous works that combine task planning and motion planning where the task planner makes a plan without considering the existence of obstacles, the motion planner cannot find the path of the arm, the task planner needs to replan many times, the motion planner needs to recheck the collision for each new task plan, and consequently the planning time can be tens of or hundreds of seconds. Because of our new heuristics, our task planner makes executable plans and saves much time.

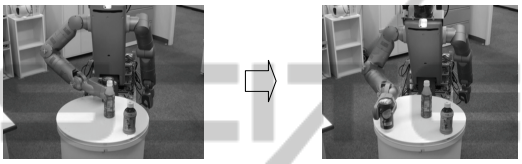

Figure 6: Moving an Obstacle.

(a) Placing an Object at the Far-Left Corner with the Right Hand

(b) Placing an Object at the Far-Right Corner with the Left Hand

Figure 7: Placing an Object inside the Fridge.

5 CONCLUSIONS

In order to make executable task plans, we introduced a new task-decomposition heuristics of HTN planning for pick-and-place manipulation. Based on the plan, the robot uses an appropriate (right or left) hand to pick and place an object without collisions, and removes some obstacles if necessary.

We tested our task-decomposition heuristics using a real robot and shelves in the room in Figure 1. We confirmed that our heuristics work as long as objects are on standard shelves (with/without walls) and within the reach of a robot hand. We also confirmed that planning time is acceptable.

Because the HTN planner considers rough location of objects and kinematic constraints of robot arms, the plan is executable for lower-level modules that use motion planners, which is different from the previous approaches that combine task planning and motion planning. Therefore, it is possible to avoid backtracking and execute the plan in real time.

REFERENCES

Baker, B. S., Jr., E. G. C., and Rivest, R. L. (1980). Orthogonal packing in two dimensions. *SIAM Journal on Computing*, 9:846–855.

Cambon, S., Alami, R., and Gravot, F. (2009). A hybrid approach to intricate motion, manipulation and task planning. *Journal of Robotics Research*, 28(1):104–126.

Choi, J. and Amir, E. (2009). Combining planning and motion planning. In *ICRA09*, pages 4374–4380.

Haspalamutgil, K., Palaz, C., Uras, T., Erdem, E., and Patoglu, V. (2010). A tight integration of task planning and motion planning in an execution monitoring framework. In *AAAI10 Workshop on Bridging The Gap Between Task And Motion Planning (BTAMP)*.

Hauser, K. and Latombe, J. (2009). Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries. In *ICAPS09 Workshop on Bridging the Gap between Task and Motion Planning (BTAMP)*.

Hayashi, H., Tokura, S., Hasegawa, T., and Ozaki, F. (2006). Dynagent: An incremental forward-chaining HTN planning agent in dynamic domains. In *Declarative Agent Languages and Technologies III*, LNAI 3904, pages 171–187. Springer.

Kaelbling, L. P. and Lozano-Perez, T. (2010). Hierarchical task and motion planning in the now. In *ICRA10 Workshop on Mobile Manipulation*.

Nau, D., Cao, Y., Lotem, A., and Mũnoz-Avila, H. (1999). SHOP: simple hierarchical ordered planner. In *IJCAI99*, pages 968–975.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Wolfe, J., Marthi, B., and Russel, S. (2010). Combined task and motion planning for mobile manipulation. In *ICAPS10*, pages 254–257.

APPENDIX

A part of this work is supported by a grant from the intelligent RT project of the New Energy and Industrial Technology Development Organization (NEDO). We also would like to thank the researchers of Yaskawa Corporation who cooperated with us in conducting experiments.