Hybrid Reasoning for Geometric Rearrangement of Multiple Movable Objects on Cluttered Surfaces

Giray Havur and **Guchan Ozbilgin** and **Esra Erdem** and **Volkan Patoglu**[1]

Abstract. We introduce a novel computational method for geometric rearrangement of multiple movable objects on a cluttered surface, where objects can change locations more than once by pick and/or push actions. This method consists of four stages: (i) finding tentative collision-free final configurations for all objects (all the new objects together with all other objects in the clutter) while also trying to minimize the number of object relocations, (ii) gridization of the continuous plane for a discrete placement of the initial configurations and the tentative final configurations of objects on the cluttered surface, (iii) finding a sequence of feasible pick and push actions to achieve the final discrete placement for the objects in the clutter from their initial discrete place, while simultaneously minimizing the number of object relocations, and (iv) finding feasible final configurations for all objects according to the optimal task plan calculated in stage (iii). For (i) and (iv), we introduce algorithms that utilize local search with random restarts; for (ii), we introduce a mathematical modeling of the discretization problem and use the state-of-the-art ASP reasoners to solve it; for (iii) we introduce a formal hybrid reasoning framework that allows embedding of geometric reasoning in task planning, and use the expressive formalisms and reasoners of ASP. We illustrate the usefulness of our integrated AI approach with a scenario that cannot be solved by the existing approaches.

1 INTRODUCTION

Natural human environments (such as refrigerator shelves, desks and tables) are often cluttered. While performing everyday chores at home, rearrangement of such clutter needs to be routinely performed either to unclutter the environment or to make space for new objects.

Geometric rearrangement planning with multiple movable objects is a challenging problem, since this task not only requires manipulation of objects lying around on the cluttered surface, but also necessitates manipulation of the new objects to be placed. Furthermore, since the order of manipulation actions matters (it may not be possible to place an object before making enough space for it), and a feasible plan may require a single object be moved *multiple times* (for instance, to swap places of two or more objects in the clutter), geometric reasoning alone is not sufficient to solve these problems; planning of manipulation actions (i.e. pick-and-place, push operations), need to be integrated with the continuous geometric problem.

Motivated by these challenges, we introduce a novel multi-stage computational method that integrates various approaches of AI. Our method consists of four stages: tentative continuous placement, discrete placement, hybrid planning, and feasible continuous placement.

[1] Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

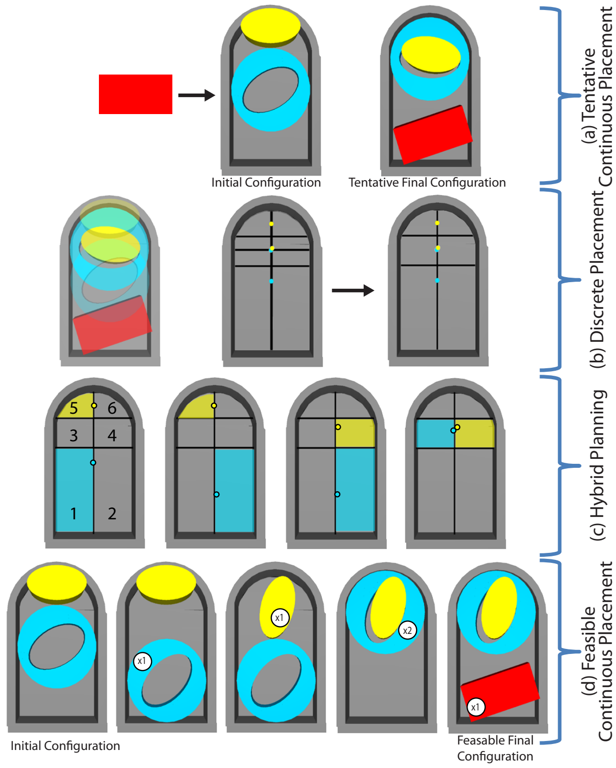

Figure 1. Tight Placement Scenario

Figure 1 illustrates our approach on a sample problem where a rectangular box is placed into a cluttered drawer; figures show perspective views of 3-D objects.

Tentative continuous placement stage finds tentative collision-free final configurations for all objects (all the new objects together with all other objects in the clutter) while also trying to minimize the number of object relocations. Note that neither the order of move actions, nor the feasibility of achieving this tentative goal configuration through these actions are considered at this stage. For a tentative continuous placement, we introduce a local search algorithm that tries to maximize the total area of the surface covered by objects. This algorithm utilizes a random sampling based collision detection to decide where to place objects on the surface, and in which orientation. Heuristics and random restarts are considered to avoid local optima. A tentative continuous placement for a tight placement sample scenario is depicted in Figure 1(a).

Discrete placement stage takes as input, the initial configurations and the tentative final configurations of all objects on the cluttered surface, and divides the surface into a *minimum* number of non-uniform grid cells. During gridization of the continuous plane, an object is allowed to (partially) span multiple grid cells as long as each grid cell contains the center of mass (or centroid) of *a single object*. For discrete placement, we formalize the optimal gridization problem in Answer Set Programming (ASP) [18, 4]—an expressive logic-based formalism, and compute solutions using the award-winning, state-of-the-art ASP solver CLASP [15, 17]. Discrete placement for the sample scenario is depicted in Figure 1(b).

Hybrid planning stage aims to find a sequence of feasible move actions (i.e., pick and push actions) to achieve the final *discrete* placement of the objects in the clutter from their initial discrete placement, while simultaneously minimizing the number of object relocations. New objects are not considered at this stage as they can be placed to their (tentative) final locations on the surface after the clutter has been rearranged. We formulate the discrete rearrangement problem for the relocated objects, as a hybrid planning problem in the spirit of [14], and use the expressive formalisms and efficient solvers of ASP to solve it. As illustrated in [13], ASP provides a formal hybrid planning framework that combines high-level representation and logic-based reasoning with low-level geometric reasoning and feasibility checks to find optimal feasible plans. So, to increase feasibility of (discrete) plans, we embed some geometric constraints on continuous placement of objects (e.g., availability of a collision-free space on the table for each manipulation action, taking into account the other objects in the clutter) in the logical formalism of ASP. An optimal task plan for the tight placement scenario is depicted in Figure 1(c).

Feasible continuous placement stage finds feasible final configurations for all objects according to the optimal task plan. Even though hybrid planning stage performs some continuous geometric checks to increase feasibility of calculated task plans, due to computationally intractable nature of the problem, embedding all continuous reasoning tasks in the domain description is not feasible. For feasible continuous placement, we introduce a local search algorithm that tries to minimize the number of collisions in an execution of the hybrid plan. This algorithm utilizes a random sampling based collision detection for objects that are manipulated, and random restarts to avoid local optima. Feasible continuous placement for the tight placement scenario is depicted in Figure 1(d).

In both continuous placement stages, our approach can utilize domain-specific constraints, such as placing an object (e.g., a monitor) in a specific area on the surface (e.g., in the corner of the table). In the hybrid planning stage, it can take into account domain-specific constraints as well: for instance, some objects may be heavy and cannot be picked by the robot but pushed.

Our framework also features several re-planning loops: In particular, if the local search algorithm can not find a feasible placement after a predetermined number of random re-initializations, then it identifies the problematic manipulation action and asks the hybrid planner to return a new discrete plan that does not involve that action. This replanning loop continues until a feasible continuous placement is found, or the hybrid planner can not return any task plan. If no task plan can be found, then the whole framework is randomly re-initialized with a new tentative continuous placement.

It is important to emphasize here the tight coupling between task planning and geometric reasoning in hybrid planning. The logic-based reasoner guides the probabilistic geometric reasoner by finding an optimal task-plan; if there is no feasible geometric solution for that task-plan then the geometric reasoner guides the task planner by modifying the planning problem with new temporal constraints. Also, we embed geometric reasoning in logic-based reasoner as described above while computing a task-plan; in that sense the geometric reasoner guides task planner to find geometrically feasible solutions.

We show the applicability of our integrated AI approach to geometric rearrangement planning, with different scenarios that cannot be solved by the existing approaches.

A longer version of this paper has appeared in Proc. of the 2014 IEEE International Conference on Robotics and Automation [16].

2 RELATED WORK

The computational problem of geometric rearrangement with multiple movable objects and its variations (like navigation among movable obstacles [23, 21], or nonprehensile manipulation under clutter [10, 11]) have been studied in literature subject to several restrictions due to their high computational complexity. Indeed, even a simplified variant with only one movable obstacle is proved to be NP-hard [25, 8]. The most common assumption that has been applied in most of the related literature is the restriction of manipulation plans to monotone plans–plans in which an object can be moved at most once. However, such a limitation causes failure when an object needs to be manipulated more than once, as seen in the scenario presented in Figure 1.

Some related works relax this monotonicity assumption by searching a manipulation solution in the robot C-space [24, 3, 5], or in the combined space of the robot and the objects altogether [2]. However, these methods are not computationally feasible for the problems with high-dimensional configuration spaces or with large number of movable objects.

Manipulation of the movable objects depends also on types of manipulation action. For instance, Cosgun et al. [7] tries to place an object on a cluttered surface, by first grasping the object, and then allowing this object to push other objects in the clutter to create a space for itself. Dogar and Srinivasa [10, 11] can accommodate both pick and place actions and non-prehensile actions such as pushes. However, these approaches are also restricted to monotone plans.

In this paper, unlike many of the related studies in literature [23, 21, 2, 19, 22], our focus is to introduce a computational framework for general manipulation plans without the restriction of monotone plans. We consider pick and place actions as well as push actions. Furthermore, our approach computes feasible and optimal task plans, thanks to hybrid reasoning aspect of our method.

3 GEOMETRIC REARRANGEMENT WITH MULTIPLE MOVABLE OBJECTS PROBLEM

The *geometric rearrangement with multiple movable objects* (GR-MMO) problem is defined by

- the geometric model $g(S)$ of a surface S that details its size and shape along with the obstacles, including non-movable objects, on it,
- a set O_C of movable objects on the (possibly cluttered) surface S and the set $g(O_C)$ of their geometric models,
- a set O_N of new objects to be placed on the surface S ($O_C \cap O_N = \emptyset$) and the set $g(O_N)$ of their geometric models,
- a set W of continuous placement constraints on objects (e.g., a monitor may be forced to be in the corner of the table),
- a set H of task planning constraints on objects (e.g., some objects may be heavy and cannot be picked by the robot but pushed),

- the initial collision-free configuration C_I of all objects in O_C on the surface S relative to $g(S)$, and
- a set A of manipulation actions $a_i(O_j)$ ($1 \leq i \leq q$) that can be used to relocate a set O_j objects ($O_j \subseteq O_C \cup O_N$) on the surface S, where $q = |A|$.

A solution to a GRMMO problem $\langle g(S), O_C, g(O_C), O_N, g(O_N), C_I, W, H, A\rangle$ consists of

- a collision-free final configuration C_F for all objects $O_N \cup O_C$ on the surface S relative to $g(S)$, and
- a feasible manipulation plan P^* given as a sequence of manipulation actions $a_i(o_j)$ that can be applied to rearrange the objects in $O_N \cup O_C$ from C_I to C_F.

Without placing any restriction on the nature of manipulation plans, we continue our discussion with two different manipulation actions that can (re)locate a single object at a time, namely, *pick-and-place* and *push* with $o_j \in O_C \cup O_N$. Note that our computational method does not depend on these assumptions and supports a diverse set of, possibly concurrent, actions including non-prehensile and uncertainty reducing manipulation actions [11] For simplicity, we also consider the surface S to be flat.

We obtain the geometric model of an object o with the help of function $g(o)$ and we abuse this notation to also operate on sets of objects. A continuous placement constraint W_o on an object o determines the area that can be sampled to find a collision-free final configuration for o. A task planning constraint H_o on an object o contains domain-specific information about o to be used during hybrid planning as a fact.

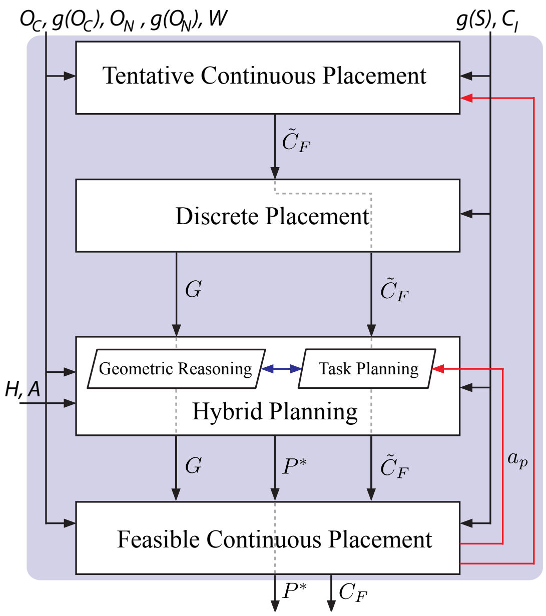

Figure 2. Computational method for geometric rearrangement with multiple movable objects on cluttered surfaces

4 AN INTEGRATED AI APPROACH TO GRMMO

We propose a novel computational method to solve GRMMO problems, that consists of four stages as depicted in Figure 2. These stages utilize local search and automated reasoning algorithms as follows.

4.1 Tentative Continuous Placement

The *tentative continuous placement* (TCP) problem is a relaxation of the GRMMO problem in which the set of manipulation actions A, their order, and task planning constraints H are not considered. A solution to a TCP problem $\langle g(S), O_C, g(O_C), O_N, g(O_N), C_I, W\rangle$ is a tentative collision-free final configuration $\tilde{C}_F$ for all objects $O_N \cup O_C$ on the surface S relative to $g(S)$.

We attack this problem by a local search strategy over search states, which are configurations of objects on the cluttered surface. Our local search algorithm aims to maximize an aggregate cost function that increases with new object placements and footprint area of new objects in the clutter, while decreasing the relocations of objects in the clutter.

First, we identify the set O_R of objects that are initially on the surface and that needs to be relocated according to W. Then, we initialize the set O_P of objects to be placed: $O_P = O_N$. With the local search algorithm, the objects in O_P are placed on the cluttered surface one by one. Essentially, due to the cost function, the objects in O_P are ordered according to their footprint area, and the search algorithm tries to place them sequentially using random sampling, starting with the largest one.

To place an object o on the cluttered surface, the surface is randomly sampled until a collision-free configuration is found or a sampling threshold is exceeded. If a collision-free configuration is found for an object o, then it is removed from O_P. If a collision-free configuration cannot be found within the sampling threshold, then the object o' that collided with o most frequently during random sampling is removed from the surface and added to the set O_P. By this way, the local search algorithm tries to avoid local optima, and allows relocation of objects that are already on the surface.

The search continues until all objects are placed or a local maxima is reached. In case of a local maxima, the search is restarted with random initialization.

Note that solution to TCP problem returns only tentative collision-free final configurations for all objects (all the new objects together with all objects in the clutter), but feasibility of achieving this tentative goal configuration through a sequence of manipulation actions cannot be guaranteed at this stage. For that, we need to decide for the order of manipulations well. This motivates the following two stages of our method: to discretize the initial/tentative configurations of objects efficiently to be able to do task planning.

4.2 Discrete Placement

A gridization of a surface with respect to a configuration of objects, like in Figure 3(a), can be obtained by iteratively dividing the midway of two closest objects' centers of mass (or centroid locations) on S with respect to horizontal or vertical axis, as in Figure 3(b). Indeed, in this way we can obtain a non-uniform grid, where each grid cell contains the centroid of a single object. However, this *naive* method of gridization may lead to a too fine grid and thus over-limit the sampling space of each object to too small grid cells. Consequently, this can prevent us from finding an existing placement solution. Another drawback of such a suboptimal gridization is the increased input size for the Hybrid Planning stage in terms of the number of grid cells. Considering the intractability of Hybrid Planning, the importance of the grid size for computational efficiency is beyond controversy. With these motivations, we aim at gridization with the minimum number of grid cells, as in Figure 3(d). For that, we define this gridization problem as an optimization problem as follows.

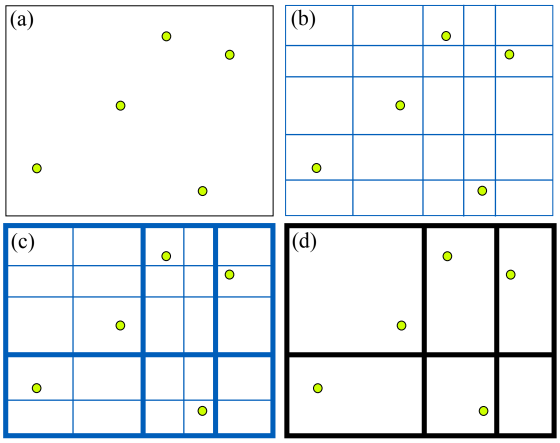

Figure 3. (a) Centroids of all objects with respect to $C_I \cup \tilde{C}_F$ are indicated on surface, (b) Program input is calculated by iteratively dividing the midway of two closest centroids horizontally or vertically , (c) Program output is emphasized, (d) Result of the grid optimization

The *discrete placement* (DP) problem is defined by

- the geometric model $g(S)$ of a surface S,
- the initial collision-free configuration C_I of all objects in O_C on the surface S relative to $g(S)$, and
- the tentative collision-free final configuration $\tilde{C}_F$ for all objects O_C on the surface S relative to $g(S)$.

A solution G to a DP problem $\langle g(S), C_I, \tilde{C}_F \rangle$ consists of

- a grid with the *minimum* number of non-uniform grid cells, such that each cell is free or contains the centroid of a single object with respect to C_I and $\tilde{C}_F$, and that at least one cell is free; and
- unique grid cell identifier for each object in $O_C \cup O_N$ with respect to C_I and $\tilde{C}_F$.

We ensure that each cell contains the centroid of a single object for two solid reasons:

- for providing a discretized input to Hybrid Planning stage to find a feasible high-level plan, and
- for guiding future continuous placement stages (Geometric Reasoning, and Feasible Continuous Placement) by limiting the sampling space for each object to a size of a grid cell, which contains at least one placement solution for the object to belong to the tentative collision-free final configuration $\tilde{C}_F$.

We solve the DP problem by mathematically modelling it in ASP as follows. We consider a suboptimal gridization for a continuous placement of objects as described above, like in 3(b), as input. Then, we find a maximal set of grid lines to remove from the suboptimal grid, to obtain a solution to the DP problem. For that, we represent the problem as a set of logical formulas (called rules) in ASP.

The rectangular suboptimal grid is defined by horizontal grid lines $y = 0, y = 1, ..., y = m$ and vertical grid lines $x = 0, x = 1, ..., x = n$. Then the optimal grid with the minimum number of non-uniform grid cells is defined by a subset of these grid lines, that satisfies the conditions about locations of objects stated in the DP problem.

We "generate" a subset of the horizontal grid lines (denoted by atoms of the form $hline(j)$) and vertical grid lines (denoted by atoms of the form $vline(j)$) by the following rules in ASP:

$$\{hline(j) : 0 \leq j \leq m\}.$$
$$\{vline(j) : 0 \leq j \leq n\}.$$

ensuring that the border grid lines are included in the subset:

$$hline(0).hline(m).vline(0).vline(n).$$

In this subset, grid cells are defined by two horizontal neighbor grid lines y_1 and y_2 (there is no other horizontal line y in between), and two vertical neighbor grid lines x_1 and x_2 (there is no other vertical line x in between) as follows:

$$\begin{aligned} & cell(x_1, y_1, x_2, y_2) \leftarrow \\ & \quad vline(x_1), vline(x_2), hline(y_1), hline(y_2), \\ & \quad \{hline(y) : y_1 < y < y_2\}0, \\ & \quad \{vline(x) : x_1 < x < x_2\}0 \end{aligned}$$

where $0 \leq x_1, x_2 \leq n$ and $0 \leq y_1, y_2 \leq m$.

The locations of objects are defined similarly, as atoms of the form $obj(x_1, y_1, x_2, y_2)$, which expresses that an object is located at the grid cell defined by the horizontal lines y_1 and y_2 and the vertical lines x_1 and x_2.

Using these definitions, we can ensure that no grid cell $cell(x_1, y_1, x_2, y_2)$ formed by the grid lines in this subset contains the centroids of any two objects, by the constraints:

$$\begin{aligned} & \leftarrow cell(x_1, y_1, x_2, y_2), \\ & 2\{obj(x_3, y_3, x_4, y_4) : x_1 \leq x_3, x_2 \geq x_4, y_1 \leq y_3, y_2 \geq y_4\}. \end{aligned}$$

Finally, we minimize the cardinality of this subset of grid lines:

$$\#minimize\,[\, vline(x) : 0 \leq x \leq n, hline(y) : 0 \leq y \leq m \,].$$

With this set of rules in ASP, we can use the ASP solver CLASP to compute a solution to the DP problem.

4.3 Hybrid Planning

Once we obtain an optimal grid and the discrete locations of the initial and the final configurations of the rearranged objects, as described in the previous section, we can find a discrete task plan for our rearrangement problem also taking feasibility checks into account.

The *hybrid planning* (HP) problem for geometric rearrangement of objects is defined by

- a grid with non-uniform grid cells such that each cell contains the centroid of a single object,
- a unique grid cell identifier for each object in $O_C \cup O_N$ with respect to C_I and $\tilde{C}_F$, and
- a set H of task planning constraints on manipulation of objects in the clutter.

A solution to an HP problem is an optimal task plan P^* with the *minimum* number of manipulation actions. To find such a plan, we use the formal framework of [13, 1] for hybrid planning that allows us to embed feasibility checks into high-level representation of actions and change by means of external predicates.

We represent hybrid planning for discrete relocations, in ASP, and use the ASP solver DLVHEX [12] to compute optimal plans as follows.

We consider fluents of the form $loc(o, c, t)$ to describe locations c of objects o at time step t. We also consider two actions of the forms, $pickPlace(r, o, c, t)$ and $push(r, o, c, t)$, to describe a robot r moving an object o to a grid cell c at time step t, by picking and placing or by pushing, respectively.

We define the direct effects and preconditions of these actions by a set of rules in ASP. For instance, the following rules express that, after a robot r picks and places an object o onto a grid cell c at time step t, the location of o becomes c at the time step $t + 1$:

$$loc(o, c, t + 1) \leftarrow pickPlace(r, o, c, t).$$

The following constraint expresses a precondition of this action: If an object o can not be grasped by the end effector of our robot r, it is not possible to pick and place the object:

$$\leftarrow pickPlace(r, o, c, t), not\ \&reachableGraspable[o, r]().$$

Here *&reachableGraspable* is an "external predicate", not a fluent or an action but a predicate whose value is calculated by an external program; it returns true if and only if the end-effector of the manipulator r can successfully reach and grasp the given object o according to kinematics and force-closure calculations of OPENRAVE [9].

The following rule expresses a precondition of the push action: a robot r cannot push an object to a location c at time step t if the volume swept by the object o from its current configuration at t towards another configuration in grid cell c, collides with other objects:

$$\leftarrow push(r, o, c, t), not\ \&pushPossible[loc, o, t]().$$

Here *&pushPossible* is an external predicate as well: it takes as input all locations of objects at time step t, and checks whether the swept volume of the object o collides with other objects using Open Dynamics Engine (ODE) [20].

If we are given some high-level constraints H, these constraints can be expressed in ASP as well. For instance, we can express that a robot r cannot pick and place heavy objects o by the constraints:

$$\leftarrow pickPlace(r, o, c, t), not\ liftable(o, r).$$

In such cases, the robot may try to push the object instead.

Also, with an optimization statement (like the one shown in the previous section), we can minimize the number of manipulation actions involved in the task plan.

With such an ASP program, we can use DLVHEX to compute a discrete task plan whose actions are feasible to the extent provided by the embedded feasibility checks by means of external predicates.

4.4 Feasible Continuous Placement

The *feasible continuous placement* (FCP) problem is a variation of the GRMMO problem, in which an optimal discrete manipulation plan $P^* = \langle A_0, A_1, \ldots, A_{n-1}\rangle$ is also provided as an input. A solution to a FCP problem $\langle g(S), O_C, g(O_C), O_N, g(O_N), C_I, W, P^*\rangle$ is a feasible collision-free final configuration C_F for all objects $O_N \cup O_C$ on the surface S relative to $g(S)$.

We solve FCP problem by a local search algorithm. In this algorithm, every state i of the search is characterized by a tuple $T_i = \langle C_0 = C_I, C_1, \ldots, C_n\rangle$ of configurations of all objects in O_C and a subset of objects in O_N. Each configuration C_{i+1} is obtained from configuration C_i by sampling the objects that are manipulated by the action A_i of the plan P^*, on the cluttered surface.

Note that when the manipulation action A_i is applied at an object within a configuration C_i by such a sampling (i.e., when picking and placing an object on to the cluttered surface, or by pushing an object on the cluttered surface) there may be collisions. The cost function for the local search algorithm is defined as the total number of configurations where such collisions are observed.

At every search state T_i, the local search algorithm decides for the next search state T_{i+1} that minimizes this cost function. If it is not possible to minimize the function (to make it 0 – no collisions), the local search algorithm restarts with a different search state by random sampling.

5 A CASE STUDY: ENFORCED SWAPPING SCENARIO

We demonstrate the applicability and effectiveness of the proposed computational method for geometric rearrangement of multiple movable objects on a sample scenario where swapping of object locations is enforced.

In this example, we introduce continuous placement constraints W to enforce a swap for final configuration of two movable objects in the clutter. A feasible swapping manipulation requires non-monotone plans, where an object must be manipulated more than once. Enforced swapping scenario is depicted in Figure 4. For this table-top setting, we enforced a swap between the printer and the PC on the desk, by defining continuous placement constraints on both objects. With these constraints in place, the tentative continuous placement stage produces a placement as in Figure 4(b), where the printer is at the former location of the PC and the PC is located at the former location of the printer.

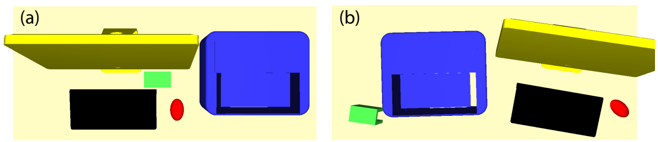

Figure 4. (a) Initial configuration, (b) Tentative final placement

Figure 5 depicts the discrete placement stage, in which an optimal grid with 9 cells is calculated. Note that, since the number of movable objects are larger in this scenario, the naive approach divides the table into 100 grid cells; hence, optimization at this stage translates to significant computational gains for the hybrid planning stage.

The hybrid planning stage finds a task plan with "pickPlace" and "push" actions as listed in Tables 1 and 2. The result of feasible continuous placement and snapshots taken during the execution of the plan are presented in Figure 6, where the swapping of PC and the printer is successfully implemented.

Table 1. Initial and final discrete placement of objects for the enforced swapping scenario

	Initial Discrete Placement	Final Discrete Placement
Blue Printer	6	4
Red Mouse	2	6
Black Keyboard	4	2
Yellow Screen	7	9
Green Desk Clock	5	1

For additional scenarios, we refer the reader to our ICRA 2014 paper [16]. For examples with dynamic simulations with the mobile manipulator CoCoA [6], we refer the reader to the URL `http://cogrobo.sabanciuniv.edu/?p=762`.

6 DISCUSSION

We have proposed a novel computational method for geometric rearrangement of multiple movable objects on a cluttered surface, where

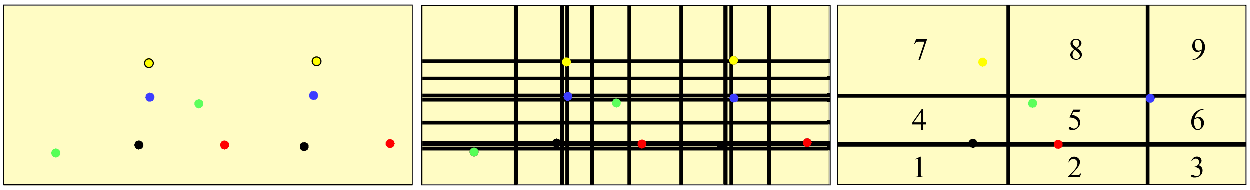

Figure 5. (a) Initial and tentative final configurations of the relocated objects (b) Non-optimal grid (c) Optimal grid with initial configurations

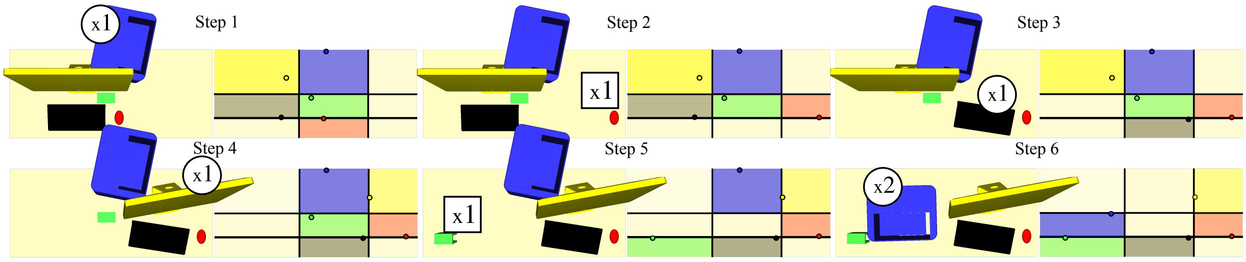

Figure 6. Snapshots taken during plan execution of the enforced swapping scenario

Table 2. A task plan for the enforced swapping scenario

Step	*Action*	*Object*	*To*
1	pickPlace	Blue Printer	Cell 8
2	push	Red Mouse	Cell 6
3	pickPlace	Black Keyboard	Cell 2
4	pickPlace	Yellow Screen	Cell 9
5	push	Green Desk Clock	Cell 1
6	pickPlace	Blue Printer	Cell 4

the objects can be relocated more than once within a plan by pick and place or push actions. In particular, we have utilized local search and automated reasoning techniques at different stages of the problem to compute feasible solutions to this problem. We have taken advantage of the expressive representation languages of automated reasoners, for describing the optimal discretization problem and for embedding results of external computations (e.g., for geometric reasoning) during task planning. We have applied the proposed computational approach to several sample scenarios that cannot be solved by the existing approaches and illustrated feasibility of our solutions by a dynamic simulation with the CoCoA mobile manipulator.

ACKNOWLEDGEMENTS

This work is supported by TUBITAK Grants 111E116 and 113M422.

REFERENCES

[1] Erdi Aker, Volkan Patoglu, and Esra Erdem, 'Answer set programming for reasoning with semantic knowledge in collaborative housekeeping robotics', in *Proc. Int. IFAC Symp. Robot Control (SYROCO)*, (2012).

[2] Jennifer Barry, Kaijen Hsiao, Leslie Pack Kaelbling, and Tomás Lozano-Pérez, 'Manipulation with multiple action types', in *Experimental Robotics*, pp. 531–545. Springer, (2013).

[3] Ohad Ben-Shahar and Ehud Rivlin, 'To push or not to push: on the rearrangement of movable objects by a mobile robot', *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, **28**(5), 667–679, (1998).

[4] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski, 'Answer set programming at a glance', *Commun. ACM*, **54**(12), 92–103, (2011).

[5] Pang C Chen and Yong K Hwang, 'Practical path planning among movable obstacles', in *Proc. of ICRA*, pp. 444–449, (1991).

[6] Gokay Coruhlu and Volkan Patoglu, 'Guvenli bilissel hizmet robotu cocoa'nin gelistirilmesi: Tasarim, modellenme ve dinamik benzetim'. Otomatik Kontrol Turk Milli Komitesi(TOK), (2013).

[7] Akansel Cosgun, Tucker Hermans, Victor Emeli, and Mike Stilman, 'Push planning for object placement on cluttered table surfaces', in *Proc. of IROS*, pp. 4627–4632, (2011).

[8] Erik D. Demaine, Martin L. Demaine, Michael Hoffmann, and Joseph O'Rourke, 'Pushing blocks is hard', *Comput. Geom.*, **26**(1), 21–36, (2003).

[9] Rosen Diankov, *Automated Construction of Robotic Manipulation Programs*, Ph.D. dissertation, CMU, Robotics Institute, August 2010.

[10] M Dogar and Siddhartha Srinivasa, 'A framework for push-grasping in clutter', *Robotics: Science and Systems VII*, (2011).

[11] Mehmet R Dogar and Siddhartha S Srinivasa, 'A planning framework for non-prehensile manipulation under clutter and uncertainty', *Autonomous Robots*, **33**(3), 217–236, (2012).

[12] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits, 'dlvhex: A system for integrating multiple semantics in an answer-set programming framework.', *Proc. of WLP*, (2006).

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

[14] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras, 'Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation', in *Proc. of ICRA*, (2011).

[15] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub, 'clasp: A conflict-driven answer set solver', in *Proc. of LPNMR*, 260–265, Springer, (2007).

[16] Giray Havur, Guchan Ozbilgin, Esra Erdem, and Volkan Patoglu, 'Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach', in *Proc. of ICRA*, (2014).

[17] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon, 'The international sat solver competitionss', *AIMAG*, **33**(1), 89, (2012).

[18] V. Lifschitz, 'What is answer set programming?', in *Proc. of AAAI*, pp. 1594–1597. MIT Press, (2008).

[19] Kei Okada, Atsushi Haneda, Hiroyuki Nakai, Masayuki Inaba, and Hirochika Inoue, 'Environment manipulation planner for humanoid robots using task graph that generates action sequence', in *Proc. of IROS*, pp. 1174–1179, (2004).

[20] Russell Smith. Open Dynamics Engine (ODE), 2006.

[21] Mike Stilman and James Kuffner, 'Planning among movable obstacles with artificial constraints', *IJRR*, **27**(11-12), 1295–1307, (2008).

[22] Mike Stilman and James J Kuffner, 'Navigation among movable obstacles: Real-time reasoning in complex environments', *IJRR*, **2**(04), 479–503, (2005).

[23] Mike Stilman, J-U Schamburek, James Kuffner, and Tamim Asfour, 'Manipulation planning among movable obstacles', in *Proc. of ICRA*, pp. 3327–3332, (2007).

[24] Jur Van Den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha, 'Path planning among movable obstacles: a probabilistically complete approach', in *Proc. of WAFR*, pp. 599–614, (2009).

[25] G. Wilfong, 'Motion planning in the presence of movable obstacles', in *Proc. of SCG*, pp. 279–288, (1988).