

# Rearrangement Competition

This document introduces a data-driven rearrangement challenge, modeling the everyday problem where many objects must be rearranged through manipulation. Where as the problem has an apparent combinatorial and geometric nature, a data-driven angle could provide a fresh angle of attack, possibly revealing intrinsic connections between spatial combinatorial problems and statistical analyses. In what follows, Section 1 explains the problem and why a learning-based approach may help. Section 3 focuses on familiarizing the readers with the three challenges with increased difficulty.

## 1 Introduction

Object rearrangement is a critical robot skill with broad applicability in logistics, home automation, and many other domains. The specific challenges discussed here focus on problem where many objects collocate in the same workspace and one object is manipulated at a time. The objective is to minimize the number of object transfers needed to complete a rearrangement task. It is assumed that a single robot arm with an end-effector is to perform the rearrangement task. The arm can reach the objects with an overhand grasp as illustrated in Fig. 1(right); it is however impossible to lift objects high enough to execute collision-free transfers. That is, as a lifted object is transferred to another location in the workspace before being settled down, the sweep volume of the lifted object's trajectory, projected on the workspace, cannot intersect other objects.

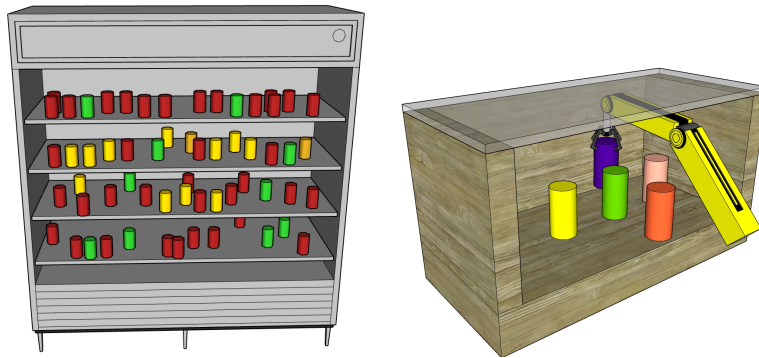


Figure 1: (left) In logistics applications, robots need to rearrange similar objects in shelves. (right) The focus is on combinatorial and geometric aspects of rearrangement when the arm can reach objects without colliding with them but cannot lift objects to guarantee they do not collide with each other.

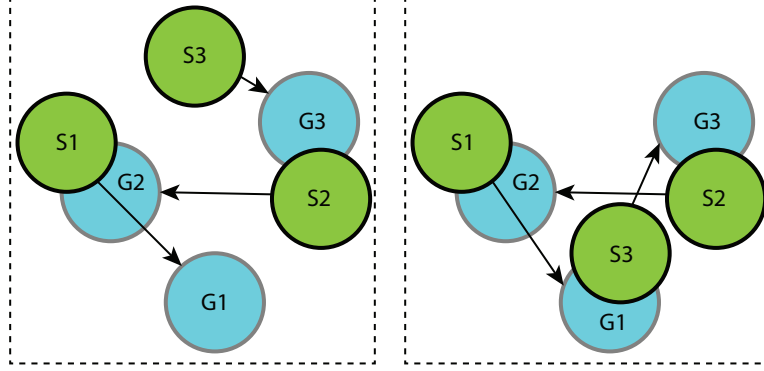


Figure 2: Two rearrangement instances where  $S_i$ (resp.  $G_i$ ) represents the start(resp. goal) pose of object  $o_i$ . [Left] An instance that can be solved with 3 actions by moving objects directly to the goal in the order of  $o_1, o_2, o_3$ . [Right] An instance that can be solved with 4 actions. By moving  $o_3$  to the top center of the environment, all the objects can move directly to the goal in the order of  $o_1, o_2, o_3$ .

Fig. 2 shows two examples of the rearrangement problems this challenge focuses on. In the instance on the left, the problem can be solved by moving objects directly to the goal in the order “ $o_1, o_2, o_3$ ” (we call such a problem monotone). In the instance on the right, the problem cannot be solved with 3 actions. At least one object needs to move to an intermediate pose before it is moved to the goal position. (we call such a problem non-monotone)

The motivation of this interesting object rearrangement challenge arises from a recent work [1], which pushes forward the understanding of the algorithmic structure of rearrangement by considering object-object collisions for uniform-shaped objects in planner setup. The work first introduces a region graph, which decomposes the workspace and classifies all continuous paths of equivalent collision sets. Based on that compact representation, an efficient monotone solver has been developed to solve monotone rearrangement problems. The monotone solver then is integrated into a high-level non-monotone framework to solve harder non-monotone instances, which needs multiple buffers. The work demonstrates the effectiveness of the methodologies in terms of 1-buffer and 2-buffer cases and thus improving the solution quality in rearrangement problems in confined space. However, there are still a bunch of problems left in the work which need to be further addressed better and motivate future work.

1. For non-monotone problems, buffers are pre-allocated as intermediate positions. However, a high-quality buffer may be generated dynamically given the current rearrangement, which adapts to specific arrangement environments to improve performance.
2. When deciding what objects to be relocated in the buffers, the selection is pretty heuristic in the work. The work generates a dependency graph based on the current arrangement and uses it to determine for each object, to what extent it is constraining or constrained. The approach can be a little bit greedy and remain unverified in harder cases where more than 2 buffers are needed.

These problems can be very computationally hard to tackle as the number of objects grows, since the size of the solution space grows exponentially. Combinatorial algorithms need to face the trade-off between computation speed and solution quality. It will be helpful if there is a learning model extracting some useful properties of the problem and leading the search of the combinatorial algorithms in the solution space.

## 2 Literature Review

In addition to the recent work [1] we mentioned above, there are an abundance of literature which study the combinatorial aspects of object rearrangement problems.

An apparently simple setup involves picking objects from tabletops, which are lifted sufficiently high before being placed back to avoid object-object collisions [2–5]. This setting is as computationally hard to solve optimally as the Travelling Salesperson Problem (TSP) when objects’ starts and goals do not overlap [2]. Overlapping starts and goals complicates the problem as it reduces to the Feedback Vertex Set (FVS) problem [6], which is possibly APX-hard. Dual arm rearrangement allows for parallelism but complicates reasoning [3]. Integer programming is often applied for deciding the object order together with motion planning [2, 3]. Optimal tabletop placement has also been approached via Answer Set Programming (ASP) [4, 5] and informed heuristics [7]. Similarly to this work, these efforts aim to minimize object grasps/pushes but address packing new items into cluttered environments; this is more akin to the unlabeled version of this work’s problem.

For more confined spaces, small monotone problems have been addressed via backtracking search [8]. A useful structure for solving problems in the general case is a “dependency graph” [9], which expresses constraints between objects given their starts and goals. Acyclic graphs indicate existence of monotone solutions. The “true” dependency graph is difficult to construct in general as all object paths must be considered. With approximations for Minimum Constraint Removal (MCR) paths [10], it is possible to practically build good dependency graphs [11, 12] to solve general instances.

## 3 Tasks

In this section, we propose three tasks aiming at unveiling properties of the rearrangement problem with learning methods.

### 3.1 Task 1: Monotonicity Detection

Monotonicity is one of the most important properties in the rearrangement problem. An instance is *monotone* if it can be solved by moving each object only once. On the other hand, problems that don’t satisfy this condition are called *non-monotone*. In Fig. 2, the instance on the left is monotone and the one on the right is non-monotone.

In this challenge, you are asked to develop a learning model that consumes start and goal arrangements of objects and classify the instances based on monotonicity.

The input offered for each task instance is a json file in the following format:

```

{
  "n": <number of objects>,
  "radius": <disk radius>,
  "height": <environment height>,
  "width": <environment width>,
  "starts": [[x0,y0], [x1,y1], ..., [xn,yn]],
  "goals": [[x0,y0], [x1,y1], ..., [xn,yn]],
  "is_monotone": <true or false>,
  "difficulty": <difficulty index>
}

```

In the input file, the "starts" and "goals" indicate the object centroids in the start and goal arrangements respectively. The "is\_monotone" entry is included to aid offline training. The "difficulty" entry indicates the difficulty level when it is solved by a combinatorial method.

### 3.2 Task 2: Object Selection

For the non-monotone instances, some of the objects need to move to an intermediate pose before moving to the goal position.

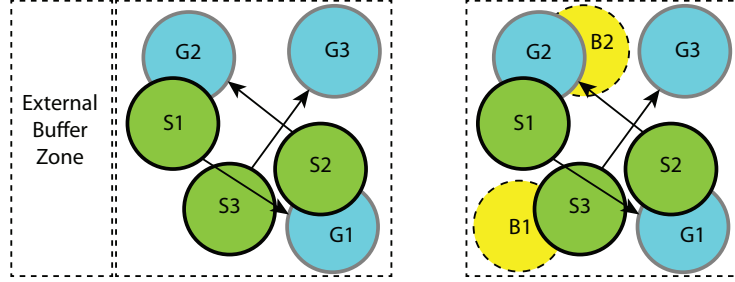


Figure 3: A rearrangement instance where  $S_i$ (resp.  $G_i$ ) represents the start (resp. goal) pose of object  $o_i$ ,  $\{B_j\}$  are a set of buffer options. [Left] In the instance,  $o_1$  blocks  $o_2$  from the goal and  $o_2$  blocks  $o_1$  from the goal as well. therefore,  $o_1$  and  $o_2$  are perturbable while  $o_3$  is not. [Right] Two buffer options for  $o_1$ . The problem can be solved by moving  $o_1$  to  $B_1$ .

In this challenge, we assume the presence of a universally accessible external buffer  $B$ . In other words, the buffer  $B$  is away from the workspace and for arbitrary pose  $p$  in the workspace, there is always a valid path between  $B$  and  $p$ . We call an object  $o_i$  *perturbable* if the instance become monotone by moving  $o_i$  to  $B$ . In Fig. 3[Left], object  $o_1$  and  $o_2$  are perturbable. After moving either of them, say  $o_1$  to the external buffer zone, the problem can be solved by moving  $o_3$ ,  $o_2$  and finally  $o_1$ . You are asked to develop a learning model to indicate the perturbability of a specific object in  $\mathcal{O}$ .

Similar to task 1, the input file is:

```

{
  "n": <number of objects>,
  "radius": <disk radius>,

```

```

    "height": <environment height>,
    "width": <environment width>,
    "starts": [[x0,y0], [x1,y1], ..., [xn,yn]],
    "goals": [[x0,y0], [x1,y1], ..., [xn,yn]],
    "object_selected" : <object index>,
    "is_perturbable": <true/false>,
    "difficulty": <difficulty index>
}

```

"object\_select" indicates the query object and "is\_perturbable" entry is the correct answer to the query, included to aid in offline training.

### 3.3 Task 3: Buffer Selection

In this challenge, we focus on instances that can be solved with  $n + 1$  actions, which means that only one of the objects needs to move to a buffer before moving again to the goal pose. Given a specific object  $o_i$  and a testing buffer location  $b$ , the goal of this task is to predict whether there is a rearrangement plan where  $o_i$  moves to  $b$  before moving again to the goal pose, and the other objects move directly to the goal. For example, the instance in Fig. 3[Right] can be solved by moving  $o_1$  to  $B_2$ .  $B_2$  are not helpful to  $o_1$ .

In the input file, in addition to the information of the environment and the objects, we further indicate a selected object  $o_i$  and the selected buffer location:

```

{
    "n": <number of objects>,
    "radius": <disk radius>,
    "height": <environment height>,
    "width": <environment width>,
    "starts": [[x0,y0], [x1,y1], ..., [xn,yn]],
    "goals": [[x0,y0], [x1,y1], ..., [xn,yn]],
    "object_selected" : <object index>,
    "buffer_selected" : [x0,y0],
    "is_valid_buffer": <true/false>,
    "difficulty": <difficulty index>
}

```

## 4 Solution Evaluation

### 4.1 Solution Format

In each challenge task, for a given instance input, the solution is a binary answer.

**Challenge 1: Monotonicity Detection** Given an instance, return "true" if the instance is monotone (i.e., each object can be directly moved from its start position to its goal position). Otherwise, return "false".

Instance File	Answer
D-0.1_n-5.0.json	true
D-0.1_n-5.1.json	false
D-0.1_n-5.2.json	false
D-0.1_n-5.3.json	true
⋮	⋮
D-0.1_n-5.49.json	true
D-0.1_n-5.50.json	false

Table 1: An example of a CSV file as the format of the solution. The left side of the CSV file lists the file name of the instance (D -  $\langle \text{density level} \rangle$  \_n -  $\langle \text{number of objects} \rangle$  \_  $\langle \text{instance index} \rangle$ .json) and the right side indicates the computed solution.

**Challenge 2: Object Selection** Given a non-monotone instance (as detected in Challenge 1) and a specified object in the instance, return "true" if that object is perturbable (i.e., the instance becomes monotone by moving that object into an intermediate position). Otherwise, return "false".

**Challenge 3: Buffer Selection** Given a non-monotone instance (as detected in Challenge 1) and a perturbable object (as detected in Challenge 2), as well as a specified buffer location, return "true" if the problem is solvable with just one additional action of moving that perturbable object to the selection buffer location. Otherwise, return "false".

Therefore, for each of the three challenge tasks, the participants are expected to submit a CSV file shown in Table 1 for each of the challenge task, as the general format of the solution. The submitted CSV file is then compared with the ground truth solution file for evaluation which has the same format as the submission.

## 4.2 Evaluation Metric

As the solution is a binary answer, the quality of the solution is reflected by how frequent the solution predict the right condition ("true" or "false"). Score is recorded as a percentage of correct classifications out of the total number of test cases.

## 5 Detailed Information to the Competition

If you are interested, please join in our competition on Kaggle. Competition link: <https://www.kaggle.com/t/a2ab98062a174f40a5544760902bf79a>

## References

- [1] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [2] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps,"

- The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.
- [3] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, “Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups,” *arXiv preprint arXiv:1810.12202*, 2018.
  - [4] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, “Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 445–452.
  - [5] A. R. Dabbour, “Placement generation and hybrid planning for robotic rearrangement on cluttered surfaces,” Ph.D. dissertation, 2019.
  - [6] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of computer computations*. Springer, 1972, pp. 85–103.
  - [7] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, “Push planning for object placement on cluttered table surfaces,” in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4627–4632.
  - [8] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, “Manipulation planning among movable obstacles,” in *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2007, pp. 3327–3332.
  - [9] J. van Den Berg, J. Snoeyink, M. C. Lin, and D. Manocha, “Centralized path planning for multiple robots: Optimal decoupling into sequential plans,” in *Robotics: Science and systems*, vol. 2, no. 2.5, 2009, pp. 2–3.
  - [10] K. Hauser, “The minimum constraint removal problem with three robotics applications,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.
  - [11] A. Krontiris and K. E. Bekris, “Dealing with difficult instances of object rearrangement,” in *Robotics: Science and Systems*, 2015.
  - [12] —, “Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3924–3931.