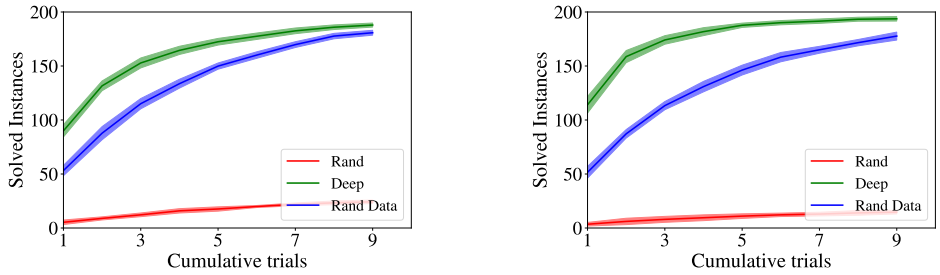


Figure 7: Cumulative Number of solved problems.

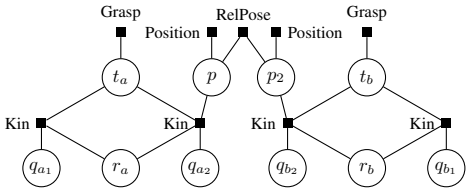


Figure 8: Constraint graph of the *Assembly* problem.

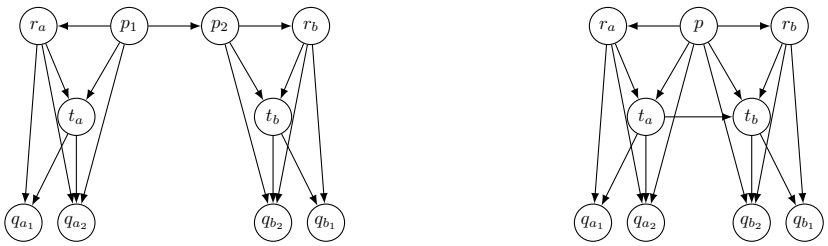
A Appendix

A.1 More results - Benchmark: Generative Model in Nonlinear Optimization

We include the plots (Fig. 7) of the cumulative number of problems solved as we increase the number of optimization trials, that were omitted due to space limitations. In the *Assembly* scenario, Deep achieves 0.57±0.04 success rate (ratio between solved and total number of problems) in one trial and 0.87±0.02 in up to 3 trials, while Rand Data achieves 0.26±0.03 in one trial and 0.56±0.02 in up to 3 trials. In the *Handover* scenario, success rates are 0.45±.03 (1 trial) and 0.76±0.02 (3 trials) for Deep; and 0.26±0.03 (1 trial) and 0.57±0.03 (3 trials) for Rand Data.

A.2 Approximate Conditional Independence

To further reduce sample complexity, we impose conditional independence between some variables in cases where this holds only approximately. Consider for example the *handover* problem (Fig. 2a), where t_a, t_b represent the relative transformation between the object and the grippers of robot A and robot B. We cannot consider that they are independent, as this would lead to a lot of gripper-gripper collisions in the handover mode-switch. Therefore, in our sampling sequence t_b depends on t_a. On the other hand, we assume conditional independence between q_{a_2} and q_{b_2} (robot configurations in the handover mode-switch) given a feasible assignment for (t_a, t_b, p, r_a, r_b). In practice, this means that our generative model (but not the subsequent optimization) ignores possible collisions between robots whose end effectors are not in collision.



(a) Assembly (b) Handover

Figure 9: Sampling networks for the *Assembly* and *Handover* problems.

Also, the positions of the robot bases r_a and r_b are assumed to be conditionally independent given the scene and the handover box position, which holds approximately in our problem instances.

Sampling networks for *Handover* and *Assembly* are shown in Fig. 9. We also include the constraint graph for the *Assembly* problem in Fig. 8, that was omitted due to space limitations.

A.3 Implementation Details

Analytical Constraint Term When training the generator network with (6), the weight β balances the constraint analytical term and the adversarial minimax objective. High values of β lead to mode collapse, while small values lead to inaccurate samples. We use the following annealing scheme, to prevent mode collapse at the beginning (when samples are inaccurate).

$$\beta(e) = \min(\beta_{\text{max}}, \beta_0 + \beta_r e) \tag{9}$$

where e is the current epoch, and $\beta_{\text{max}}, \beta_0$ and β_r are user-defined values for the maximum, initial and increase rate of β.

The analytical constraint term $||\phi(x;\tau)||^2$ with the squared penalty explodes when samples are inaccurate. We use a smooth $l1$-loss (Hueber loss) that combines l_1 loss for big errors, and l_2 for small errors,

$$L(\phi_i) = \begin{cases} 0.5\phi_i^2/\alpha & \text{if } |\phi_i| < \alpha \\ |\phi_i| - 0.5\alpha & \text{otherwise} \end{cases} \tag{10}$$

where $\alpha \in \mathbb{R}$ is a user-defined value.

Architecture Generator and critic (discriminator) networks have a similar architecture. First, the image input is encoded with 3 convolutional layers and a final fully connected layer. In the generators, the encoding is concatenated with the low dimensional conditioning (if any) and the noise, and passed to a fully connected network with 4 hidden layers, that outputs a vector (sample). In the critics (discriminators), the image encoding is concatenated with the low dimensional conditioning and the sample, and the network (with 3 hidden layers and dropout regularization) outputs a scalar. Both convolutional and fully connected layers use Leaky Relu as activation function.

In the current implementation, we do not use shared modules. The image encoding could be shared or trained simultaneously, at least between similar sampling operations (for example, by using the same encoding for all samplers of robot configurations). This could considerably reduce the computational time of training and inference.

Training Each sampler of the sequence is trained individually, with the corresponding marginal distribution and analytical constraints. We have observed a slight performance degradation when all modules are evaluated in a sequence, as opposed to individually with the ground-truth marginal conditioning. One of the reasons is that the output distribution of a generative model does not match exactly the training input of the following module. To alleviate this issue, we will further investigate the addition of noise during training, using out-of-distribution samples and retraining the full sequence.

A.4 Collection of Samples

In this section, we display samples generated by our generative models in the three robotic scenarios. These images are also shown in the supplementary video.

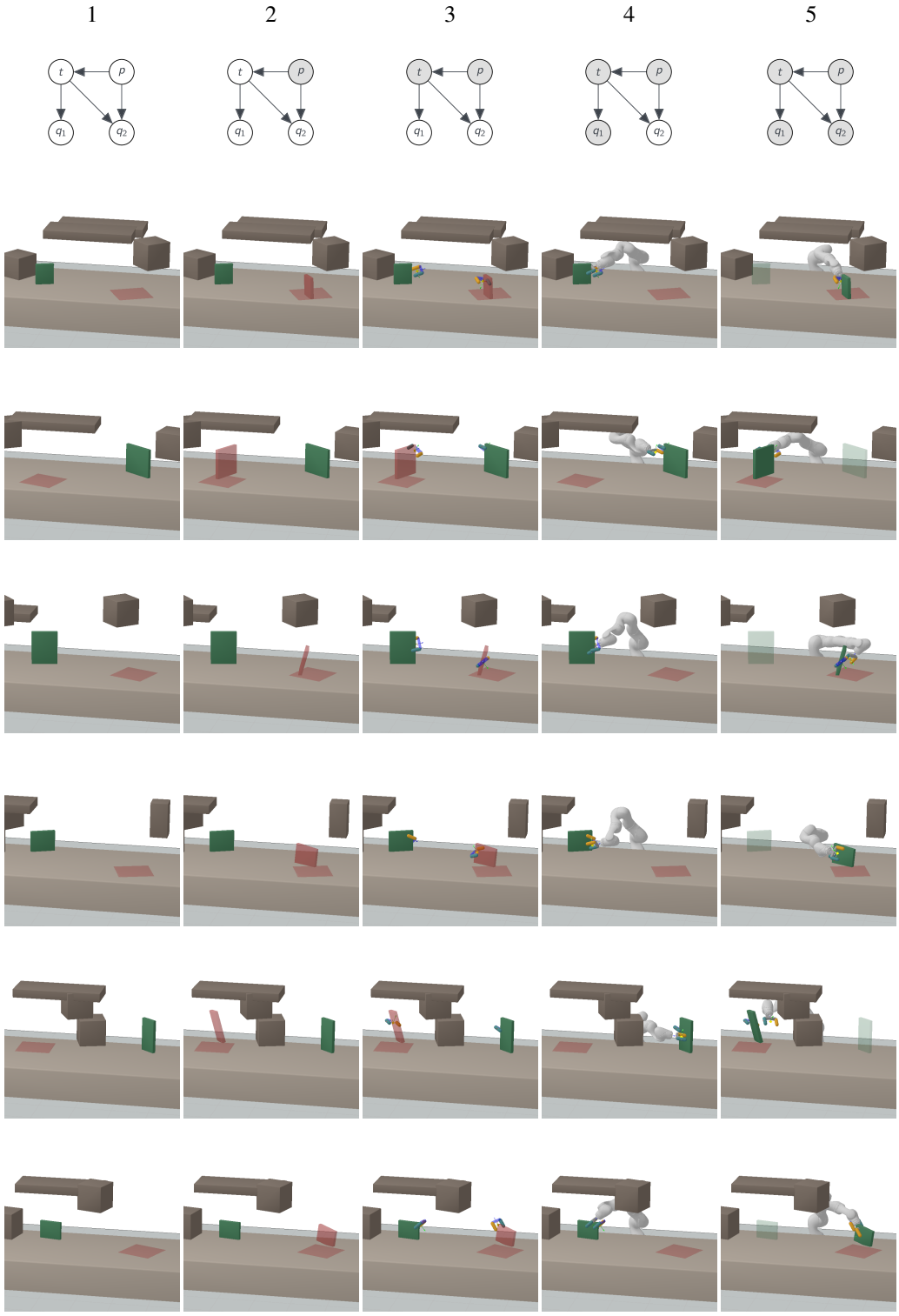


Figure 10: Sequence of the 4 generative models that compose the sampling network for the *pick-and-place* problem. Each row corresponds to a sample. Column 1 shows the problem instance, with the initial object position (green) and goal region (red). We first sample the box on the goal region (column 2). We then generate the grasp, that is represented with a flying gripper (column 3). Finally, we sample the robot configurations at pick and place (columns 4 and 5).

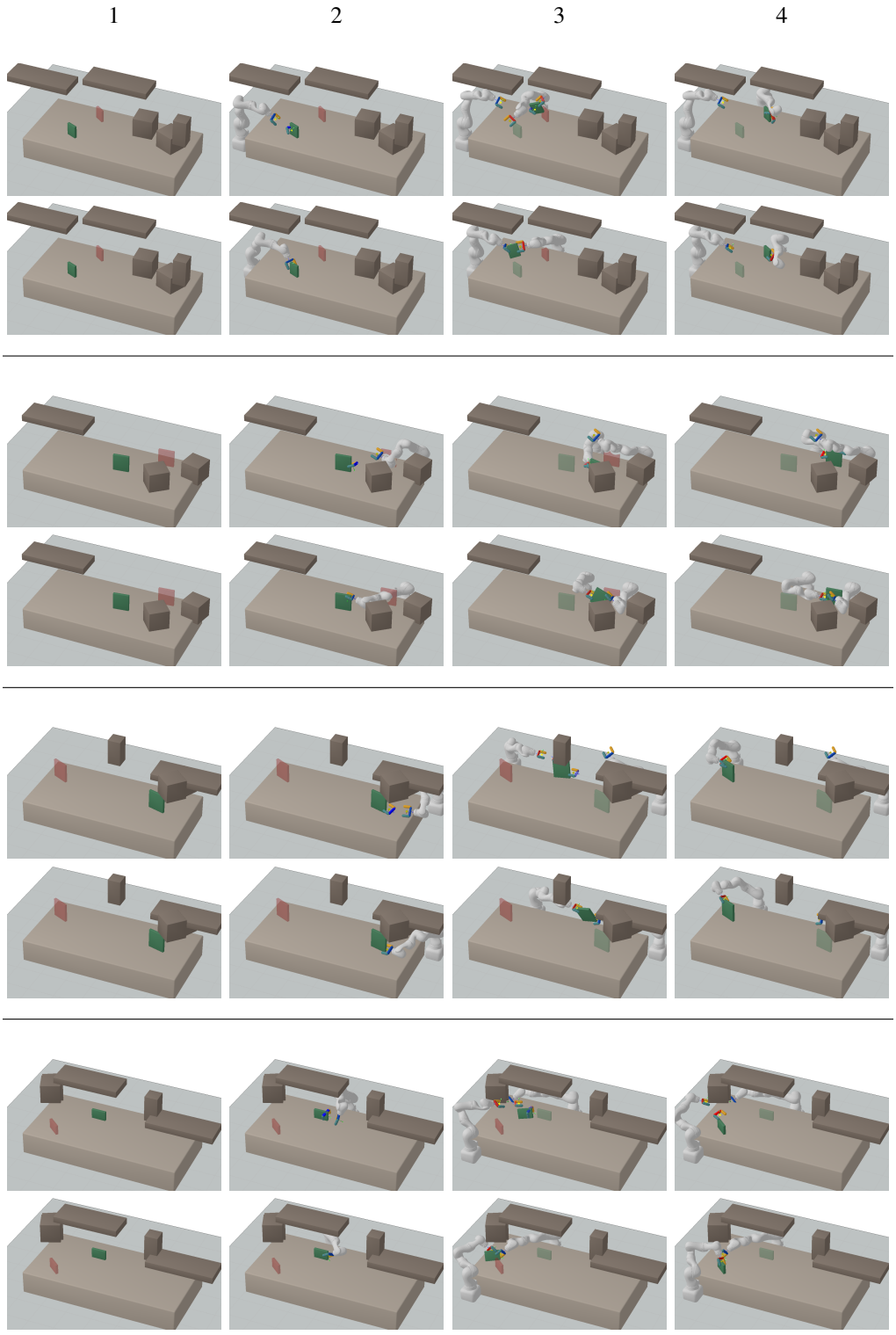


Figure 11: Samples in the *Handover* problem (Part 1). We display a sample by showing the three mode-switches of the sequence (each row is a sample). In each block of two rows, we show the output of the generative model (above) and the optimized solution (below). Column 1 shows the problem instance. In column 2, robot A picks the object, and the flying gripper represents the grasp. Column 3 displays the handover, with the handover box position in green. In column 4, robot B places the object in the goal position.

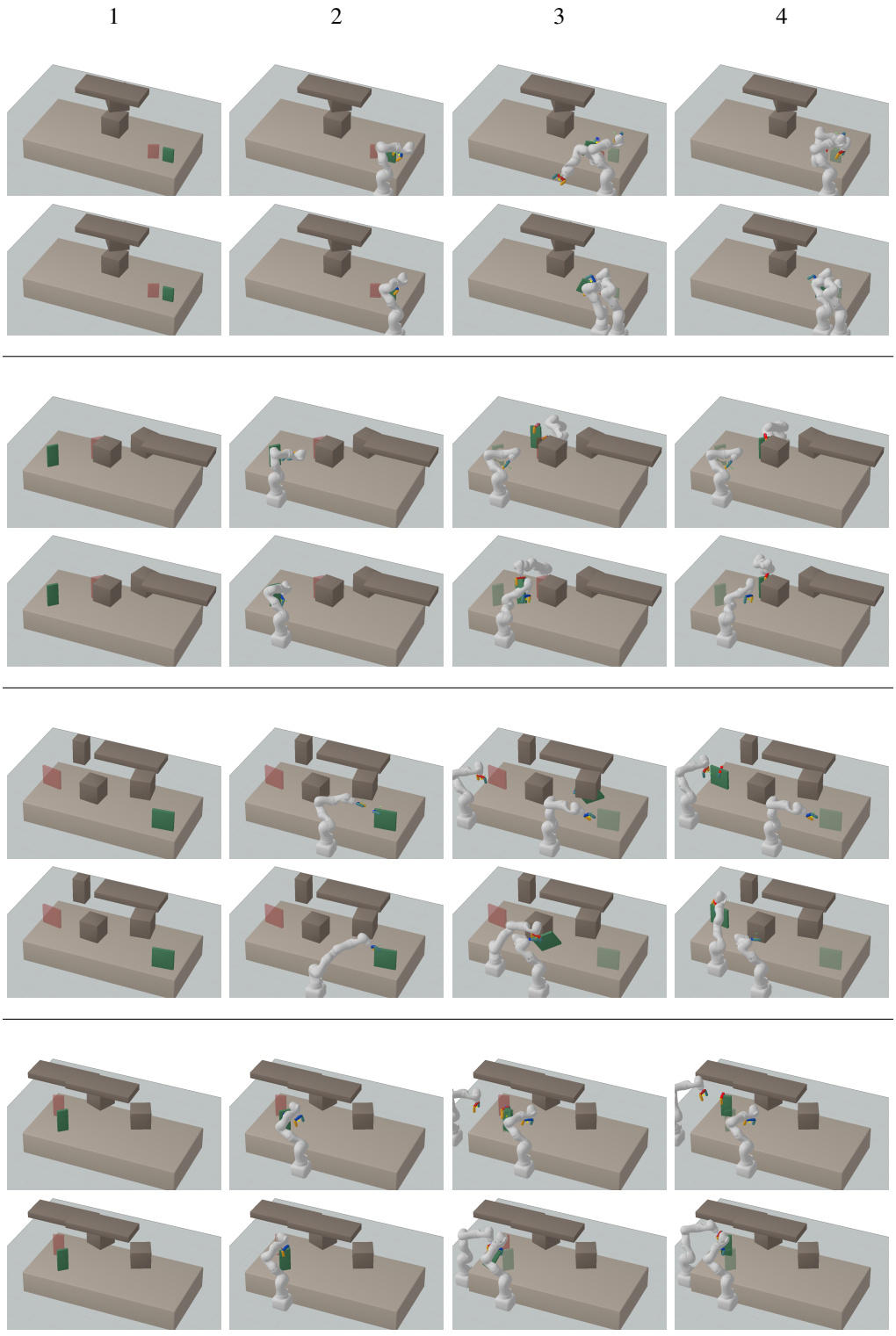


Figure 12: Samples in the *Handover* problem (Part 2). See caption of Fig. 11.

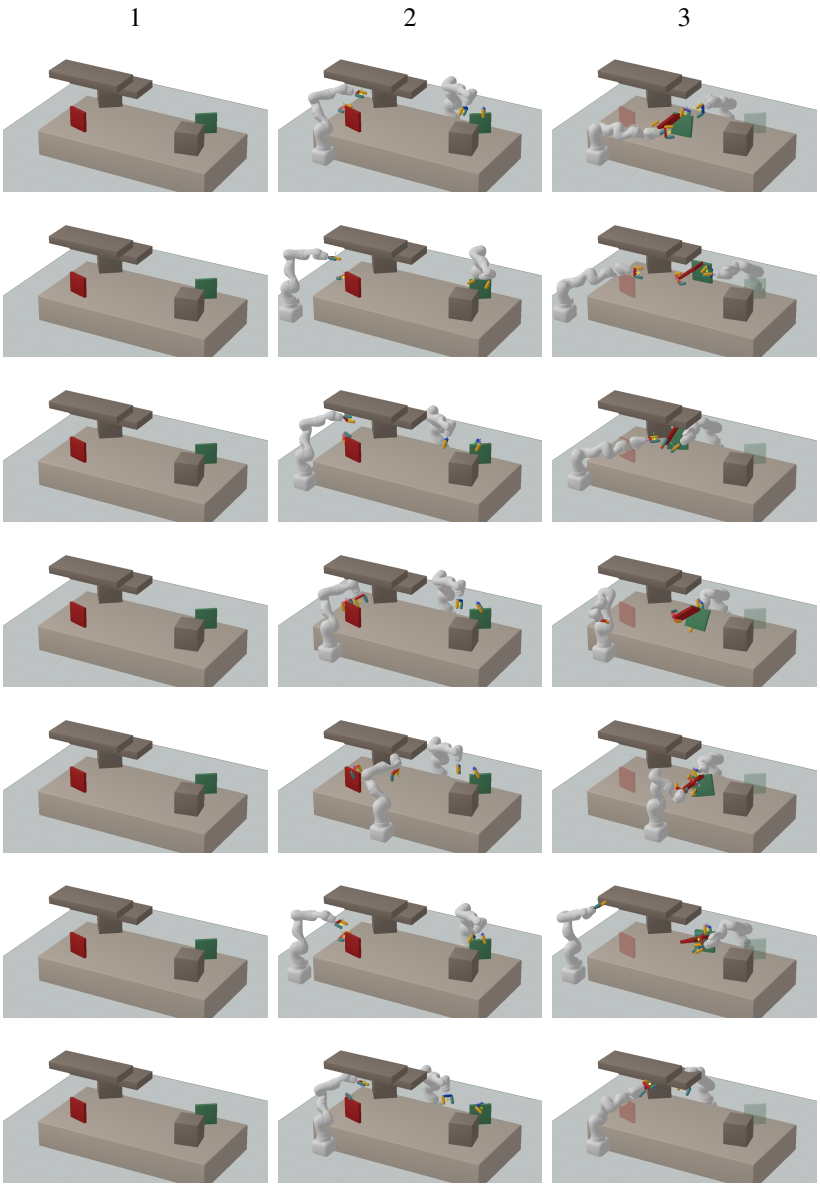


Figure 13: Samples in the *Assembly* problem (Part 1). We display a sample (only the output of the deep generative model, not the optimized solution) by showing the two mode-switches of the sequence. Note that, in this figure, the problem instance (column 1) is the same for all the samples. Column 2 shows the pick mode-switch, where each robot picks one object. Column 3 shows the assembly (not unique, T-shape like) of the green and red boxes.

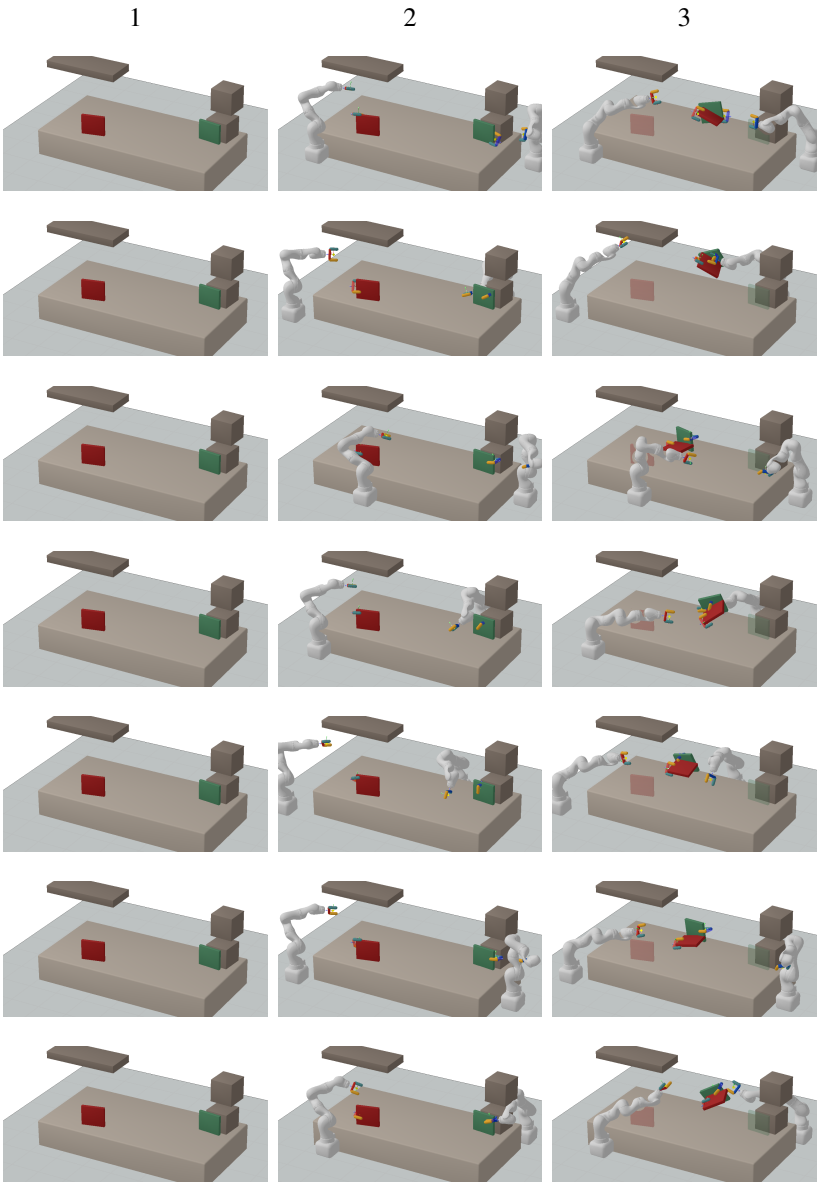


Figure 14: Samples in the *Assembly* problem (Part 2). See caption of Fig. 13.