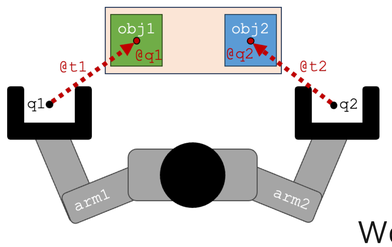
ICRA IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION

Poster Contacts: Caelan Garrett (cgarrett@nvidia.com), Fabio Ramos (ftozetoramos@nvidia.com)

NVIDIA



ScheduleStream: GPU-Accelerated Multi-Arm Task and Motion Planning & Scheduling

Caelan Garrett | Fabio Ramos | NVIDIA Research | Seattle Robotics Lab (SRL)

Website: schedulestream.github.io | Code: https://github.com/NVlabs/ScheduleStream | Video: https://youtu.be/0LyTPmAXaQY

TL;DR: ScheduleStream is a framework for manipulation planning & scheduling with GPU-accelerated multi-arm robot constraints and samplers

1. ScheduleStream Language

Multi-Arm Robots Need Parallel Plans

Existing manipulation planning approaches predominantly produce sequential plans, leaving arms idle while others act

Planning vs. Scheduling

- **Plans**: a sequence of instantaneous actions
- **Schedules**: a set of timed, durative actions executed in parallel

Applications

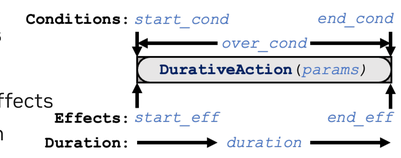
- Multi-arm industrial fabrication and assembly
- Automated demonstration generation in simulation
- Grounding agentic plans with multi-arm constraints and costs

Representational Contributions

- First **action language** for mixed discrete-continuous planning & scheduling with procedural constraints and samplers
- **Batched** constraints and samplers for GPU acceleration

Extend STRIPS/PDDL Actions

1. Continuous parameters
2. Start & end conditions & effects
3. Continuous action duration



Python Durative Actions: Multi-Arm Move Action

```
move = DurativeAction(
    # robot arm ?arm, start conf ?q1, trajectory ?t, end conf ?q2
    parameters="?arm ?q1 ?t ?q2",
    start_condition=[
        Motion("?arm ?q1 ?t ?q2"),  # valid motion constraint
        At("?arm") == "?q1",  # ?arm is currently at start conf ?q1
    ],
    start_effect=[At("?arm") <= "?t"],  # ?arm is now along trajectory ?t
    over_condition=arm_obj_collisions + arm_arm_collisions,
    end_condition=[],
    end_effect=[At("?arm") <= "?q2"],  # ?arm is now at end conf ?q2
    min_duration=Duration("?t"),  # Duration of trajectory ?t elapses
)
```

Batched GPU-Accelerated Collision Constraints

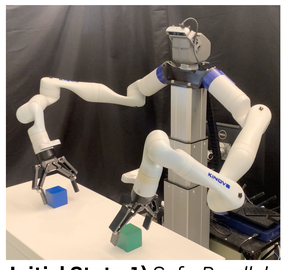
```
arm_obj_collisions = [~ObjCollision("?t", Pose(o)) for o in [obj1, obj2]]
arm_arm_collisions = [~ArmCollision("?t", At(a)) for a in [arm1, arm2]]
```

2. ScheduleStream Algorithms

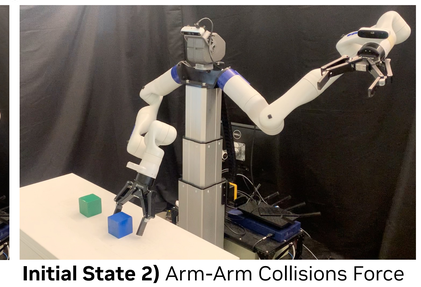
Arm-Arm Collisions Impose Temporal Constraints

Collision and kinematic constraints limit **which** arms can do **what** manipulation **when.** Solution viability depends on schedule timing

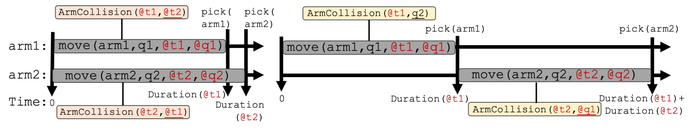
Goal: "Blue in Right Hand, Green in Left Hand"



Initial State 1) Safe *Parallel* Arm Schedule Exists



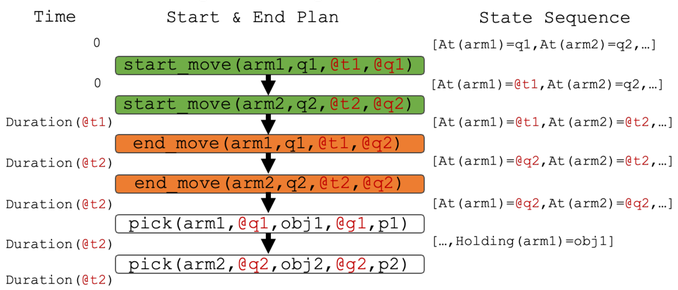
Initial State 2) Arm-Arm Collisions Force *Sequential* Arm Schedule



Algorithmic Contributions

- Lazy weighted A* search over action **start** and **end** events
- **Batched** parameter sampling and constraint satisfaction

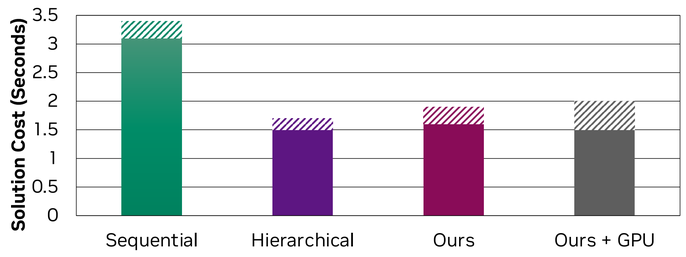
Reduce Scheduling to Planning a Sequence of Start & End Actions



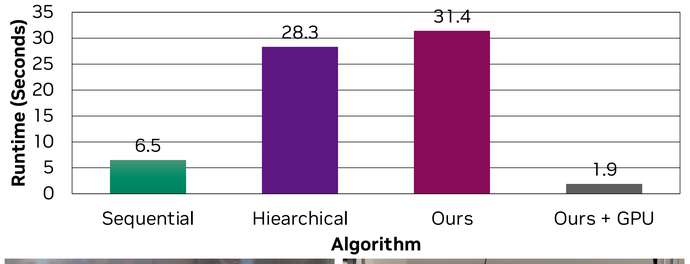
3. Experimental Results

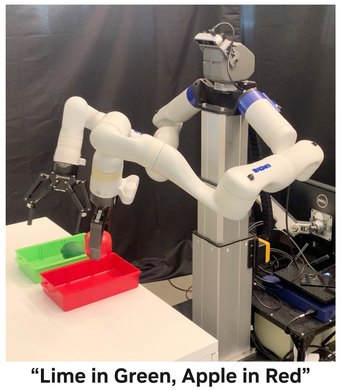
Analyze Simulation Solution Cost (Duration) and Algorithm Runtime

Solution Cost: Simultaneous Multi-Arm Motion

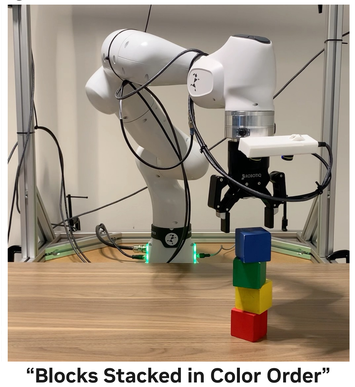


Algorithm Runtime: GPU-Accelerated Planning





"Lime in Green, Apple in Red"



"Blocks Stacked in Color Order"