AUTON: Labeled Failure-and-Recovery Data for Multi-Robot Fleets from Typed Fault Injection in Simulation

***Abstract*—AUTON generates simulated data of multi-robot fleets failing and recovering, for teams that train and test fleet software and robot world models. Its input is one site file: robot types, a site map and a task list. It runs simulated shifts on the site and injects eight typed faults, among them a robot dying mid-task, a dropped radio link, a blocked path and two robots sent head-on into a single lane. A fault changes the simulated robot, link or site and is never reported to the coordinator, which sees only symptoms such as missing heartbeats. Every coordinator choice is logged under a sequence number, and each shift returns a label recording what failed, how and how quickly it was detected, which decision answered it, and whether the fleet finished the work. Where the coordinator's own decision is at the root of a deadlock or of duplicated work, the label names it and the shift is replayed with only that decision changed. A campaign of 6,000 shifts on kinematic bodies (a five-robot earthworks yard and a seven-robot warehouse, four coordinator presets) injected 8,998 faults and logged 330,915 decisions; 72.7% of shifts completed, and in 3,676 of 4,366 replays changing the named decision removed the incident. Seven fault schedules were rerun on PhysX articulations in NVIDIA Isaac Sim 6.2.0: four agree with their kinematic twins on outcome, tasks done, detection path and recovery, with makespans 0.6–11.1% longer, and three diverge. Two of those diverge in traffic at single-lane gates, one through a physical wedge that kinematic bodies cannot produce. The allocator and precedence logic under test were validated earlier in Webots under a per-robot ROS 2/Nav2 stack (greedy allocation at most 3.3% behind Hungarian on the kinematic backend; task fraction 0.99 ± 0.03 against the abstraction with a 2.5–$3.7\times$ makespan inflation; precedence DAGs of depth 1, 7, 9 and 15 completing $8/8$, $8/8$, $7/8$ and $0/8$ over eight physics reps per tier). All evidence is simulation: nothing has run on a physical robot, and the fault models are not fitted to field failure data.**

I. Introduction

A fleet coordinator writes its own logs. When a robot goes silent, the log holds what the coordinator concluded (the robot is down), which can differ from what happened (the radio link dropped and the robot kept working). The log also cannot say what a different decision would have produced, because the shift ran once. Teams that train coordination policies or world models on fleet data, and teams that must test a fleet-software release before it reaches a site, need both missing pieces: ground truth about each failure that does not depend on the coordinator's diagnosis, and a way to rerun the same shift with one choice changed.

AUTON generates that data in simulation. The input is one site file with three parts: the robot types and fleet, a site map, and a task list with precedence. AUTON runs simulated shifts on the site and injects faults from a catalog of eight types (Table I), among them a robot dying mid-task, a dropped radio link, a blocked path, and two robots sent head-on into a single lane. Each shift returns a label that records what failed, how and how fast the coordinator detected it, which numbered decision answered it, and whether the fleet finished the work. We see two uses: training data for coordination policies and world models [35], and a test suite that a team runs before pushing new fleet software to a site.

The labels depend on a separation the engine enforces in code. *Truth* is where every body is, what is broken and what work has physically been done. *Belief* is what the coordinator holds, built only from telemetry that arrived. A fault changes truth, and the coordinator is not told. One tested schedule shows what this buys. A warehouse robot loses its radio link for 40 s under a coordinator that presumes a robot down after 2.5 s of silence. The robot keeps working. The coordinator hands its task to a second robot, and when the link returns two robots hold one task. The label records the detection path (missing heartbeats), the misdiagnosis (believed down; in truth alive with its link down), a duplicate-work incident, and the root decision, a numbered `PRESUME_DOWN`. Replaying the shift with that single decision suppressed removes the duplicate work, and the label records that as well (§IV-C).

The evidence comes from three simulation backends, named with every result: kinematic bodies for the 6,000-shift campaign (§V); PhysX articulations in NVIDIA Isaac Sim [31] for seven rerun schedules, of which four agree with their kinematic twins and three do not (§VI); and Webots [25] with a full ROS 2/Nav2 stack per robot for the earlier validation of the allocator, precedence rule and planner that the fault library now exercises (§VII, §VIII). Nothing here has run on a physical robot.

Contributions.

1) **An input contract**: one site file holding robot types, a site map with named single-lane passages, and a task list with precedence (§III-A).
2) **A fault library** of eight typed faults acting on the drive, a tool, the radio link or the site, with injected truth kept out of everything the coordinator reads (§IV-A).
3) **A numbered decision log and a label schema** covering detection path and latency, diagnosis correctness, the answering decision, displaced tasks and recovery (§IV-B).
4) **Attribution by one-decision replay**: for deadlocks and

duplicate work with a coordinator decision at the root, the shift is rerun with that decision changed and the label reports whether the incident still occurred (§IV-C).

5) **A campaign of 6,000 labeled shifts** on kinematic bodies across two sites and four coordinator presets (§V).
6) **An Isaac Sim backend** (PhysX articulations, RTX frames, per-frame ground truth) and a seven-schedule comparison with the kinematic bodies, which also exposed a routing bug (§VI).
7) **The earlier Webots/Nav2 validation**: a six-allocator comparison, a measured abstraction gap, a reactive layer for spatial constraints, and precedence DAGs of depth 1 to 15 soaked at eight reps per tier (§VII, §VIII).

II. Related Work

Failure data and fault injection. Field studies document how ground robots fail physically [32], and fault detection and diagnosis for robots has its own survey literature [33]. Chaos engineering injects failures into running software services to test whether they recover [34]. AUTON applies injection to a simulated fleet, where the injected fault is known exactly and can be withheld from the system under test. ALLIANCE showed behavior-level fault-tolerant cooperation [11]; our subject is the labeled record of such recoveries and of the failures to recover. World models are learned from recorded rollouts of an environment [35], and a fleet world model needs rollouts that include failures. We have trained no model on this data.

Routing, allocation and avoidance. Classical MAPF routes homogeneous agents on a known graph; lifelong variants add online task assignment [4], bounded-suboptimal solvers scale it [5], and downstream work bridges plans to kinodynamic motion [6]. The fleets here are differentiated (excavate $\rightarrow$ transport $\rightarrow$ place), which moves the hard problem from routing to allocation and sequencing. The Gerkey–Matarić taxonomy [1], its iTax extension [2] and temporal-constraint variants [3] frame that space, and dynamic settings use market-based methods (MURDOCH [7], TraderBots-style systems [8], CBBA [9]). The coordinator under test uses a greedy nearest-compatible scheme in the capability-constrained ST-SR region. At the execution layer, ORCA splits avoidance symmetrically in velocity space [15], buffered Voronoi cells (BVC) need only neighbor *positions* [12], and control barrier functions constrain the control input [13]. Symmetric schemes livelock when a slow agent meets faster ones, which motivates asymmetric variants such as weighted BVC [14].

Construction and collective robotics. Single-machine construction autonomy is maturing: HEAP trenches and builds dry-stone walls autonomously [16], [17], and publication counts in the field are rising [29]. Surveys trace earthmoving toward equipment teaming [18] and multi-robot allocation [19]; digital-twin/BIM scheduling exists but assumes human execution [28]. In TERMES [20], an offline compiler turns a target structure into local rules that robots follow by local sensing alone, on robots and blocks co-designed for the task [22], with no run-time coordinator. The rules inherit a correctness guarantee [21]: any execution order produces the structure and cannot deadlock. AUTON's single coordinator offers nothing comparable on ordering. The guarantee rests on homogeneous agents and fungible bricks, whereas AUTON's fleets are capability-typed and a required capability may sit on one machine, so the failure of interest is loss of a *non-redundant capability* (§VIII-B).

III. Inputs and the Shift Engine

A. The site file

One YAML file per site holds what a user supplies. *Robot types and fleet*: per type a capability set, a speed and a body radius; per robot a type, a start position and a parking spot. *Site map*: bounds, fixed obstacles as rectangles, and the single-lane passages (aisles or gates) that labels refer to by name. *Task list*: per task a required capability, a location, an optional drop-off, a nominal work time and the tasks it requires. The task list is the capability-typed precedence DAG formalized in §VII-A. The loader rejects a file it cannot run (a task site inside an obstacle, a task no robot type can perform), and the engine rejects a fault schedule it cannot honour, so no shift runs with a fault silently dropped.

Two sites ship. `construction_yard` is the Webots arena of §VIII at the same 1:20 scale. A spoil berm splits it into a laydown side and a dig side joined by two single-lane haul gates. Five robots (two excavators at 0.11 m/s, two haulers and a surveyor at 0.22 m/s) work 16 tasks: survey, dig, haul spoil, inspect, deliver, place footings. `warehouse_aisles` is a 36×22 m warehouse at full scale, where six rack blocks leave five 1.8 m aisles too narrow for two robots to pass. Seven robots (four AMRs, two forklifts and a scout, at 1.1–1.6 m/s) work 24 tasks: picks to pack stations, inbound pallets, put-aways and cycle counts. Each fleet has exactly one robot able to inspect.

B. Truth, belief and numbered decisions

The engine advances in 0.1 s ticks and keeps three records apart. *Truth*: where every body is, what is broken, what work has physically been done. *Belief*: what the coordinator holds about each robot, built only from telemetry that arrived. *Decisions*: every coordinator choice, numbered in order (`d0001`, `d0002`, ...) with its kind (dispatch, reassign, route, reroute, presume down, take out of service, declare deadlock, break deadlock and others), its inputs, and the other agents and tasks it could have picked.

Each tick applies any due fault to truth, delivers telemetry from robots whose link is up, lets robots sense unmapped obstacles, runs the coordinator's watchdogs, allocates ready tasks to idle robots, resolves traffic, then moves bodies and advances work. A shift ends COMPLETE when every task is done, HALTED when every remaining task is blocked and nothing in the world can still change, or TIMEOUT.

A shift writes four streams: `events.jsonl`, the mission log the coordinator could have written itself, in which a test asserts no fault identifier or injection record appears; `decisions.jsonl`; `truth.jsonl`, for the evaluator only (injections, health changes, physical completions);

TABLE I

THE FAULT CATALOG (FAULTLIB/CATALOG.PY). A FAULT CHANGES THE SIMULATED ROBOT, LINK OR SITE. THE COORDINATOR RECEIVES NO FAULT RECORD AND SEES ONLY THE SYMPTOM IN THE LAST COLUMN.

Fault	What is changed	What the coordinator can see
Robot death	drive and telemetry cut together	heartbeats stop; the body stays as an obstacle
Immobilized	drive only	position stops changing while the robot reports driving
Degraded speed	drive speed limit	progress falls behind the nominal rate
Capability loss	one tool	a tool-fault report on the next attempt to use it
Intermittent dropout	drive and telemetry for a window	a heartbeat gap, then the robot reappears
Comms loss	telemetry and commands, both ways	heartbeats stop while the robot keeps working
Blocked path	a new rigid obstacle on the site	a robot senses it within range and line of sight
Aisle contention	two relocation orders into one lane	both robots report being blocked by the other

and `timestep.csv`, a 10 Hz pose trace. A shift is fixed by its site file, policy preset, environment seed and fault schedule. The seed jitters each robot's speed by up to 6% and each task's work time by up to 15%, and identical inputs reproduce identical logs and labels.

C. The coordinator under test

The coordinator in these campaigns is AUTON's own baseline. It allocates with the greedy nearest-compatible rule of §VII-C, imported from the allocator module that §VIII-C compares against Hungarian and CBBA, and plans routes with the A* grid planner of §VII-B. Four watchdogs supply its view of failure: a heartbeat timeout (6 s) after which a silent robot is presumed down and its task returns to the pool; a no-progress timeout (8 s) for a robot that reports driving without moving; a blocked-too-long escalation (45 s); and a slow-progress flag (below 60% of nominal speed for 5 s). A wait-for cycle that persists for 6 s is declared a deadlock and broken by sending the lowest-priority robot in the cycle to a pull-over point. The coordinator has no lane reservations, so two robots can meet head-on in an aisle with no fault injected.

Its constants live in one policy object with four presets: *baseline*; *naive* (no deadlock breaker, downed bodies not stamped on the shared map); *impatient* (2.5 s heartbeat timeout); and *patient* (20 s). A campaign mixes the presets so the data holds shifts that recover and shifts that do not. Another team's fleet software would take this object's place. We have not yet connected one.

IV. FAULT LIBRARY AND LABELS

A. Eight typed faults

Table I lists the catalog. Five faults act on a robot, one on the radio link, one on the site and one on traffic. Each catalog entry also states what its model leaves out and how the Isaac Sim backend applies it. The coordinator can learn of a fault through seven paths: the four watchdogs of §III-C, a tool-fault report, a robot sensing an obstacle within range and line of sight, and a wait-for cycle.

Detection through symptoms produces cases that a kill signal cannot. A tool failure stays latent until the robot next tries to use the tool. A 3 s dropout passes unnoticed under the 6 s heartbeat timeout, while a 15 s one is detected, misdiagnosed as a robot down, and reconciled when the robot rejoins. A radio drop and a death are indistinguishable to the coordinator until the link returns. Aisle contention is staged: the injector releases two relocation orders at opposite mouths of one lane, and whether the meeting becomes a deadlock, and whether the fleet gets out of it, is left to the policy under test.

B. The label record

Each shift yields one record under the schema `auton.shift_label/1`. The *outcome* block gives the mission result, tasks done, makespan, delay against a fault-free twin, and the robots lost. Each *fault* entry gives the injection, the state the robot and its task were in, whether the fault *manifested* (a broken tool that is never needed again changes nothing), the detection (path, latency from onset, the detecting decision and, for a presumed-down call, the believed and the true state), the decisions that answered it, the tasks it displaced, and recovery. A fault counts as *recovered* only if it manifested, every task it displaced was finished, every deadlock linked to it cleared, and the shift completed. In a shift with two or three faults, one fault's recovery therefore depends on the others. Each *incident* entry (deadlock, duplicate work, false alarm, stranded robot and, on PhysX only, physical wedge) gives the agents, the place and the root cause. The record closes with a plain-language summary. One from the yard campaign reads: "Surv1 lost its radio link for 12 s at 60 s while driving to T1. The coordinator caught it 6.0 s after onset from missing heartbeats. It presumed the robot down; the robot was alive with its link down. Recovered: yes (T1 finished by Surv1). All 16 tasks finished."

C. Which decision: attribution by replay

Two incident types can have a coordinator decision at the root. For a deadlock, the engine names the later of the route commitments that put the robots into the shared lane. For duplicate work, it names the decision that returned the task to the pool while its first owner was still working, in practice a `PRESUME_DOWN`. Naming is a hypothesis, so the engine tests it. The shift is replayed from the start with the same seed and fault schedule and one override: the named route is planned around the lane, or the presume-down is suppressed so the coordinator keeps waiting. The label then records whether the incident recurred (for a deadlock, the same robots within 60 s of the original time; for duplicate work, the same task), together with the replay's outcome and makespan change. At most two incidents are replayed per shift. Deadlocks staged by the injector are labeled as injected contention and are not replayed.

A replay that avoids the incident shows that the named decision was necessary for it in that shift, relative to the one alternative tried. It does not show that the decision was the

TABLE II
FAULT-LIBRARY CAMPAIGN BY FAULT TYPE (KINEMATIC BODIES, SEED 42, FOUR PRESETS MIXED). DETECTED AND RECOVERED SHARES ARE OVER MANIFESTED FAULTS. A FAULT COUNTS AS RECOVERED ONLY IF THE WHOLE SHIFT COMPLETED.

Fault	Inj.	Manif.	Det. (%)	Median detect (s)	Recov. (%)
Construction yard (3,000 shifts, 4,464 faults)					
Robot death	695	695	97.7	6.0	52.5
Immobilized	435	363	97.8	8.1	51.5
Degraded speed	489	399	87.7	5.1	88.2
Capability loss	480	282	100.0	2.0	59.6
Intermittent dropout	473	409	89.7	6.0	80.4
Comms loss	648	621	96.6	6.0	83.3
Blocked path	628	599	100.0	0.0	76.5
Aisle contention	616	553	100.0	6.0	73.8
Warehouse (3,000 shifts, 4,534 faults)					
Robot death	739	739	99.9	6.0	44.2
Immobilized	453	412	93.9	8.1	50.5
Degraded speed	435	382	81.7	5.2	79.3
Capability loss	522	425	100.0	2.0	64.2
Intermittent dropout	471	443	84.4	6.0	74.3
Comms loss	639	632	95.1	6.0	78.0
Blocked path	648	520	100.0	17.8	29.0
Aisle contention	627	602	100.0	6.0	73.9

TABLE III
THE SAME CAMPAIGN BY COORDINATOR PRESET (KINEMATIC BODIES): SHIFTS COMPLETED, DEADLOCKS CLEARED/DECLARED, AND DUPLICATE-WORK INCIDENTS. THE BASELINE RAN 1,200 SHIFTS PER SITE AND EACH OTHER PRESET 600.

	Construction yard			Warehouse		
Preset	Compl. (%)	Deadlocks cleared	Dup. work	Compl. (%)	Deadlocks cleared	Dup. work
Baseline	85.1	1367/1379	92	76.2	1003/1019	131
Naive	43.5	46/545	44	38.5	37/555	84
Impatient	84.3	640/649	54	77.7	581/592	60
Patient	84.8	560/565	18	75.8	425/430	24

only contributor. Replays in which the incident still occurs stay in the data, and §V reports both counts.

Tests. Twenty unit tests check what a label consumer relies on. Identical inputs give identical logs and labels. Each fault type is detected through the path its catalog entry names. Over 24 sampled shifts stepped tick by tick, covering both sites and all four presets, no fault identifier reaches the mission log, no task stays with two reachable robots for more than one tick, no task completes before its prerequisites, and no task is credited without having been physically done. A duplicate-work label traced to a presume-down decision survives the replay.

V. CAMPAIGN: 6,000 LABELED SHIFTS

Setup (kinematic bodies). We ran 3,000 shifts on each site with base seed 42. Each shift draws one, two or three faults (probabilities 0.6, 0.3, 0.1). Types are weighted 1.6 for robot death, 1.4 each for comms loss, blocked path and aisle contention, and 1.0 for the other four. Injection times are uniform between 6% and 72% of the site's median fault-free makespan (373.4 s in the yard, 187.2 s in the warehouse). Presets rotate baseline, baseline, naive, impatient, patient, so the baseline runs 1,200 shifts per site and each other preset 600. Every shift has a fault-free twin under the same preset and environment seed (40 seeds per preset), and each campaign's manifest records its command and site-file hash. The fault mix is our sampling choice and estimates no field failure rate.

Outcomes. The yard campaign injected 4,464 faults and logged 144,651 decisions; 2,297 of its 3,000 shifts completed (76.6%) and 703 halted. The warehouse campaign injected 4,534 faults and logged 186,264 decisions; 2,066 shifts completed (68.9%) and 934 halted. No shift timed out. Together that is 8,998 faults and 330,915 decisions, with 4,363 of 6,000 shifts complete (72.7%). A completed shift finished a median 4.9 s later than its fault-free twin in the yard and 2.6 s later in the warehouse (90th percentiles 84.6 s and 28.0 s).

By fault type (Table II). Shares are over manifested faults: in the yard 480 capability-loss faults were injected and 282 manifested. Detection is complete at both sites for capability loss, blocked path and aisle contention, and between 93.9% and 99.9% for robot death, immobilized and comms loss. It is lowest for intermittent dropout (89.7% yard, 84.4% warehouse) and degraded speed (87.7%, 81.7%): by construction, a dropout shorter than the heartbeat timeout and a slowdown that keeps more than 60% of nominal speed trip no watchdog. Median time to detect follows the policy constants: 6.0 s for the heartbeat faults and the wait-for cycle, 8.1 s for the no-progress watchdog, 2.0 s for the tool report. Blocked path is the exception, at 0.0 s in the yard, where a robot was within the six-radius sensing range on the tick the box appeared in at least half the cases, and 17.8 s in the 12 m warehouse aisles. All 1,201 detected comms-loss faults and all 741 detected dropouts were misdiagnosed as a robot down: a heartbeat timeout cannot tell silence from death.

Recovery separates the fault types more than detection does. Faults that leave the robot able to finish its own work recover most often (degraded speed, comms loss and dropout, 74.3–88.2%). Faults that remove a robot or a tool recover least (capability loss, death and immobilized, 44.2–64.2%). Blocked paths split by site: 76.5% recover in the yard, where a second gate offers a way round, and 29.0% in the warehouse, where the box sits at an aisle's midpoint beside pick and put-away faces and about half the sampled blockages never clear. In the 40-label sample shipped with the results, all five unrecovered warehouse blockages list a task sited in the blocked aisle as never completed. Every share in this column is also lowered by the naive preset, since a fault counts as recovered only if its shift completes.

By preset (Table III). The baseline, impatient and patient presets complete 84.3–85.1% of yard shifts and 75.8–77.7% of warehouse shifts; naive completes 43.5% and 38.5%. Naive lacks the deadlock breaker and the stamping of downed bodies, and the deadlock counts show the first at work: the baseline cleared 1,367 of 1,379 declared deadlocks in the yard, naive 46 of 545. The patient preset has the fewest duplicate-work incidents (18 and 24 in 600 shifts). The impatient preset (54 and 60) sits close to the baseline's rate (92 and 131 in 1,200 shifts).

Incidents and attribution. The labels hold 5,734 deadlock



Fig. 1. One RTX frame from the blocked-gate schedule in Isaac Sim 6.2.0 (PhysX bodies, 1:20 scale): two haulers, with the laydown stacks behind. Vehicle meshes are visual only; collision uses boxes and spheres. Every frame is saved with a ground-truth record of each robot's pose, state and fault flags.

incidents (3,138 yard, 2,596 warehouse), of which 4,659 cleared. In 4,150 the root is a coordinator route decision with no staged contention behind it: without lane reservations the baseline coordinator sends two robots through one gate or aisle from opposite ends as a matter of course, so deadlocks outnumber the 1,243 injected aisle-contention faults several times over. The labels also hold 507 duplicate-work incidents, all rooted in a decision, 1,551 stranded-robot incidents and 7 false alarms. Of the decision-rooted incidents, 4,366 were replayed with the root decision changed. In 3,676 (84.2%) the incident did not recur: 2,000 of 2,512 in the yard (79.6%) and 1,676 of 1,854 in the warehouse (90.4%). In the other 690 it still occurred, and for those the named decision is a candidate that the test did not confirm.

VI. Isaac Sim Backend and Two-Backend Comparison

The engine reaches bodies through a small interface: each 0.1 s tick it hands every robot a polyline and a speed cap and reads back a pose. The campaign backend moves each body along its polyline exactly. A second backend runs the same coordinator, fault injection, logging and labeling against PhysX articulations in NVIDIA Isaac Sim 6.2.0 [31], and records RTX-rendered frames with one ground-truth record per frame: every robot's pose, state and flags, and the active faults.

Environment. Isaac Sim 6.2.0 from pip on Python 3.12, headless on an AWS `g6e.2xlarge` (8 vCPU, 64 GiB, one NVIDIA L40S), Ubuntu 24.04.5, NVIDIA driver 620.70.06. PhysX runs the TGS solver at 120 Hz; the coordinator ticks at 10 Hz.

What is ported. The stage is authored from the site file: a triangle-mesh terrain with a static collider, berm ridges and task markers. Robots are floating-base articulations with a differential drive that keeps the Webots drive base (wheel radius 0.033 m, track 0.16 m, 2 kg, the same collision boxes); the vehicle meshes are visual only. A pursuit controller turns the polyline and speed cap into wheel velocity targets, so wheel slip, contact and slope are in the loop and a robot arrives where physics puts it. Death and immobilization zero the wheel targets on a stiff velocity drive, degraded speed scales them, a blocked path spawns a rigid box, and the radio and dropout faults mask telemetry.

TABLE IV

Seven fault schedules on the construction yard (environment seed 1), each run on the kinematic bodies (K) and on PhysX articulations in Isaac Sim (P). Preset in parentheses. 4 of 7 schedules agree on outcome, tasks done, detection path and recovery; † marks the 3 that do not.

Schedule		Outcome	Tasks	Makespan (s)	Detected by	Detect (s)	Rec.	Lost
No fault (base.)	K	compl.	16/16	374.9	n/a	n/a	n/a	none
	P	compl.	16/16	381.1	n/a	n/a	n/a	none
Hauler death (base.)	K	compl.	16/16	415.7	heartbeat	6.0	yes	Haul1
	P	compl.	16/16	418.0	heartbeat	6.0	yes	Haul1
Head-on (base.)†	K	compl.	16/16	409.2	wait cycle	6.0	yes	none
	P	halted	9/16	n/a	wait cycle	6.0	no	Surv1
Blocked gate (base.)	K	compl.	16/16	406.6	sensing	1.1	yes	none
	P	compl.	16/16	451.8	sensing	2.4	yes	none
Radio drop (impat.)	K	compl.	16/16	373.5	heartbeat	2.5	yes	none
	P	compl.	16/16	380.9	heartbeat	2.5	yes	none
Lost drive (base.)†	K	halted	11/16	n/a	no progress	7.9	no	Exc1
	P	halted	4/16	n/a	no progress	8.1	no	Exc1
Head-on (naive)†	K	halted	9/16	n/a	wait cycle	6.0	no	Surv1
	P	compl.	16/16	514.7	undetected	n/a	no	Exc1

What is not. ROS 2 and Nav2 are not in this loop. The coordinator runs in-process against PhysX poses, with no lidar, AMCL, costmap or MPPI; the Webots stack of §VIII remains the only evidence for those. The excavator boom and hauler bed do not articulate, digging and dumping are timed dwells, and there is no soil model or terrain deformation, so these runs say nothing about the ODE pit-slope artifact of §VIII-F. The warehouse site builds, but only the yard has recorded runs.

Comparison (Table IV). Seven fault schedules on the yard, all with environment seed 1, were run on both backends: a fault-free shift; a hauler death at 130 s; a staged head-on in the south gate under the baseline and again under the naive preset; a 150 s blockage of the north gate; a 40 s radio drop under the impatient preset; and a lost drive on an excavator. The comparison checks four fields: mission outcome, tasks done, detection path, and whether the fault was recovered. Four schedules agree on all four. In those, PhysX makespans are longer by 0.6% to 11.1% (2.3 to 45.2 s), heartbeat detections land at the same 6.0 s and 2.5 s because they are timers, and the blocked gate is sensed after 2.4 s against 1.1 s.

Three schedules disagree. Two turn on traffic in a gate. In the baseline head-on, the kinematic shift completes 16/16. Under PhysX the surveyor, ordered to pull over to clear a deadlock with an excavator, wedged against it with their centres 0.375 m apart and no drive fault injected. The no-progress watchdog pulled it from service, and because the surveyor is the only inspector the shift halted at 9/16. The label records a *physical wedge*, an incident kinematic bodies cannot produce. In the naive head-on the direction reverses. Kinematically the staged pair deadlocks, the surveyor is pulled from service and the shift halts at 9/16. Under PhysX the staged pair does not deadlock; an unstaged excavator–hauler deadlock strands Exc1 instead, and the shift still completes

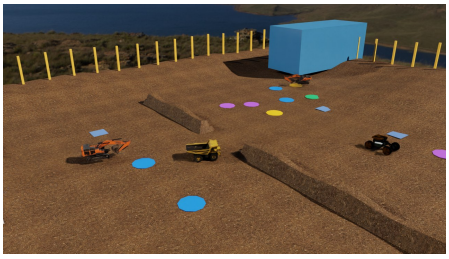


Fig. 2. One RTX-rendered frame of the construction yard in Isaac Sim. The robots are PhysX articulations with the Webots vehicle meshes as visuals; berm segments and task markers come from the site file; the container and fence posts are dressing.

16/16 at 514.7 s. In the lost-drive schedule both backends halt with the same robot lost and the same detection path (7.9 s and 8.1 s), but PhysX completes 4 tasks where the kinematic run completes 11. We have not traced that difference.

Our reading is narrow. The detection path matched in five of the six schedules that carry a fault; the exception is the naive head-on, which never formed under PhysX. Outcomes that depend on which two robots meet in a single-lane gate, and when, did not carry over: PhysX bodies accelerate, turn in place and can touch, so arrival times shift and a different pair can meet. A kinematic label is exact for the kinematic model, and for gate traffic it does not predict the PhysX outcome of the same schedule. Across the seven PhysX runs the worst body tilt was 6.5° and the closest approach between two robot centres 0.368 m; the runs executed at 1.90 to 2.35 times real time with frame recording on.

A bug the PhysX run exposed. Running a schedule on PhysX exposed a routing bug that the kinematic runs had hidden. A detour planned around a dead body could pass just inside the distance at which a robot still counts as blocked by that body, so the robot waited beside it until the coordinator pulled it from service. The keep-out disc used to route around a body is now wider than the body by the same margin the yield rule uses, so a detour the planner accepts is one the robot can drive, and a robot with no way round hands its task back. Both backends were rerun after the fix. The yard campaign's manifest carries the fix commit; the warehouse manifest and the Isaac Sim run metadata record no commit.

VII. The Coordinator Under Test

The coordinator was built and measured first as a ROS 2 stack for construction fleets, described here and validated in §VIII (full write-ups under `results/`). Three pieces carry over into the fault library: the allocator module and the A* grid planner, both imported unchanged, and the ready-set precedence rule, reimplemented in the engine. The Nav2 stack, the reactive layer of §VII-E and the rescue mechanisms of §VII-C do not, and the watchdogs and traffic rules of §III-C are separate code. A fault in these two sections is a kill signal or an embodiment failure the stack detected itself; none of the eight typed faults of §IV-A appear.

A. Problem formulation

Let $A = \{a_1, \ldots, a_n\}$ be the fleet and $\mathcal{C}$ the capability types (Excavate, Place, Transport, Inspect); agent a_i carries a static set $C_i \subseteq \mathcal{C}$. Each task $t_j \in T$ has a required-capability set $\rho(t_j) \subseteq \mathcal{C}$, a location $\mathbf{x}(t_j) \in \mathbb{R}^2$, a duration $d(t_j) > 0$ and prerequisites $P(t_j) \subseteq T$, which induce an acyclic graph $G = (T, E)$. Agent a_i is *eligible* for t_j iff $\rho(t_j) \subseteq C_i$, and t_j is *ready* once $P(t_j)$ is complete. Σ collects the constraints on concurrent execution (collision zones, dynamic keep-out regions, approach/exit paths, occupancy limits) and is part of the problem statement; $\Sigma = \emptyset$ produces the avoidable collisions of §VIII-A. The *achievability set* $\mathcal{A}(M, E)$ holds the structures realizable from material M in environment E whatever the fleet. We treat it as an external input, since *how well* a fleet builds an achievable structure is the coordination problem: find an assignment and schedule $f : T \to A$, $S : T \to \mathbb{R}_{\geq 0}$ subject to **(C1)** capability feasibility, $\rho(t_j) \subseteq C_{f(t_j)}$; **(C2)** precedence, $S(t_j) \geq S(t_k) + d(t_k)$ for all $(t_k, t_j) \in E$; **(C3)** no agent executing two tasks at once; and **(C4)** spatial feasibility w.r.t. Σ; minimizing the makespan $C_{\max} = \max_j \big(S(t_j) + d(t_j)\big)$. Our working hypothesis was that fleet performance is bounded more by (f, S) and the handling of Σ than by per-agent execution fidelity.

B. Architecture

The ROS 2 [23] stack has three layers. The coordination layer is one node, `mission_control`, owning the task pool, precedence graph, fleet model, allocation and fault recovery. The navigation layer is, per robot and namespace, an action server implementing `nav2_msgs/NavigateToPose`. The substrate is a kinematic simulator or a Webots 3D physics world [25]. `mission_control` issues cancelable NavigateToPose goals, consumes odometry, ground-truth poses, action feedback and the `/fault` channel, and publishes `/mission_status`. It does not know how motion is realized or which substrate runs beneath it.

For the discrete-event campaign (§VIII-B) navigation is a compact A* [30] planner over an inflated occupancy grid with line-of-sight smoothing. For full physics it is a complete Nav2 stack [24] per namespace: AMCL localization against the as-built grid, layered costmaps, a NavFn global planner, and an MPPI local controller tuned per role (excavator capped at 0.11 m/s; haulers and surveyor at 0.22 m/s) under a periodically replanning behavior tree. `mission_control` is byte-for-byte identical over both. A terrain server streams occupancy patches as the world changes (a completed dig stamps a lethal pit disc; a spoil mound or disabled robot gets a keep-out) and each costmap replans around them: the dynamic keep-out part of Σ.

C. Allocation, fault recovery and rescue

The deployed fleet has four robots: Exc1 {Excavate, Place}; Haul1, Haul2 {Place, Transport}; Surv1 {Inspect}. Excavate and Inspect are single-agent capabilities. Each tick, `mission_control` computes the ready set $T_{\text{ready}} = \{t \in T_{\text{pending}} \mid P(t) \subseteq T_{\text{done}}\}$ and matches each

idle agent a_i, in priority order, to its nearest eligible ready task,

$$t^* \;=\; \arg\min_{t \in T_{\text{ready}},\ \rho(t) \subseteq C_i} \big\|\mathbf{x}(a_i) - \mathbf{x}(t)\big\|_2,$$

dispatching a NavigateToPose goal to $\mathbf{x}(t^*)$. The rule is greedy and myopic and we claim no optimality for it.

On a fault for agent a_i executing t, the coordinator cancels the goal, returns t to the pending pool, marks a_i unavailable and re-dispatches when an eligible agent idles. No task is dropped, and with no spare capacity an orphaned task waits. A robot that wedges against geometry or falls into a pit is downed through the same path with a distinct cause code.

Downing is irreversible within a mission, so four mechanisms try rescue first: a *failure-gated nudge* (after a streak of nav aborts with zero ground-truth motion and a correct localization belief, the reactive layer relocates the robot a short distance); a one-shot *sole-capability grace* for the last live holder of a still-required capability; a *corridor-park release* for a robot whose return-to-depot drive livelocks; and a *ground-truth goal-drive*. The last exists because on the deepest DAGs the excavator's accumulating pits and spoil mounds are absent from the AMCL reference map, scan-matching drifts the belief metres off truth, and the planner routes against a pit it cannot see past. When the nudge-and-detour ladder is exhausted, the robot drives to the task goal by ground truth, once per leg, releasing at the ground-truth arrival gate (§VIII-D). A robot still wedged after that is downed.

D. Physics substrate and the render-only plugin

In Webots, a `site_plugin` with Supervisor powers subscribes to `/mission_status` and renders coordination state into physical effects: a pallet snapped onto a hauler's bed, a dig stroke that carves terrain, staged blocks placed. It publishes ground-truth poses, assigns no work and influences no decision, which lets one binary drive the physics-free and full-physics worlds. The height-field (procedural Perlin terrain [27]) is only ever lowered, and displaced material appears as a spoil mound sized to the excavated volume. Once the digger has backed clear, carve deltas mirror into the collision height-field and the pit is a hazard: stamped into costmaps, guarded by the reactive layer and, if a robot drives in, detected as a HOLE-FALL fault. Each dig is checked against ground truth, and every dig in the reported runs passes.

E. Realizing Σ: priority buffered Voronoi cells

The collision-zone part of Σ is a decentralized reactive layer *beneath* the NavigateToPose interface. Each robot stays inside its Voronoi cell shrunk inward by a safety radius r_s, its buffered Voronoi cell [12]. For robot i at p_i and neighbor j at p_j, the retracted bisector gives one linear constraint on the next position x:

$$n_{ij}\cdot(x - p_i) \;\le\; \tfrac{1}{2}\|p_j - p_i\| - r_s, \qquad n_{ij} = \tfrac{p_j - p_i}{\|p_j - p_i\|},$$

and the controller projects its desired one-step displacement onto the intersection of these half-planes. Buffered cells never overlap, and the layer needs only neighbor positions and priorities at 10 Hz.

TABLE V
FAULT RESILIENCE, FULL FLEET (DISCRETE-EVENT, 60 TRIALS/CELL, SEED 42; MEAN ± 95% CI). RECOVERY LATENCY RUNS FROM THE FAULT TO COMPLETION OF THE DISPLACED TASK, WHERE A CAPABLE BACKUP EXISTS.

Scenario	Compl. (%)	Makespan (s)	Recov. lat. (s)	Blocked
No fault	100	79.7 ± 3.8	n/a	0.0
Hauler fault	100	100.2 ± 6.4	46.3 ± 5.1	0.0
Exc. fault	5	67.3 ± 3.7	26.8 ± 5.1	1.6 ± 0.2
Cascade	85	99.8 ± 7.1	54.3 ± 8.1	0.2 ± 0.1

Pure reciprocal avoidance livelocks: fast haulers and the slow excavator yield to each other into a comoving stand-off until bodies touch. A discrete right-of-way in the spirit of weighted BVC [14] breaks the symmetry: robot i constrains against neighbor j only when $\pi_j \ge \pi_i$. A static order starves its bottom, though. The surveyor, lowest in priority, stalled chronically and was downed for "no progress" while obeying the order exactly. Under *priority aging*, an agent holding a live goal with no net headway for a fixed interval temporarily boosts its priority by a large constant until it drives clear. A *stall escape* handles an agent stuck against static geometry (spoil flanks, pit rims): a lethal-adjacent agent motionless for a few seconds drives out along a scored collision-free bearing. Soak testing exposed three defects in it. Escapes now carry the aging boost (in the deep-tier baseline **74% of escape commands were zero-velocity holds**; now 2–4%). A chosen escape lane is committed until walked, stale or invalidated. Linear speed is capped above a pit's keep-out ring, after which pit-falls dropped to one across the 24-run sweep.

VIII. PRIOR VALIDATION IN WEBOTS WITH ROS 2/NAV2

A. Full-physics demonstration and the $\Sigma = \emptyset$ baseline

Webots physics. The deployed mission runs the four-robot fleet on nine tasks (2 EXCAVATE, 3 TRANSPORT, 2 PLACE, 2 INSPECT). At t=10.5 s a fault is injected on Haul1 mid-TRANSPORT: its goal is canceled, the carried pallet returns to source, and the task re-dispatches to Haul2 24 s later when it next idles. The same run exhibits the $\Sigma = \emptyset$ failure. Haul2's path crosses the stationary excavator, the bodies collide, and with Haul1 already down no hauler remains, so `mission_control` returns the blocked task to the pool and finds no eligible agent. This run is the baseline for the reactive layer (§VIII-E).

B. Discrete-event campaign

Discrete-event simulator. A campaign of **900 trials** (60 per cell) sweeps fault scenario, fleet composition, obstacle density and mission size on a headless simulator that implements `mission_control`'s logic exactly over the same A* planner (Table V).

A fault on a *redundant* capability is absorbed at 100% completion for ≈26% more makespan and a ≈46 s capacity-bounded recovery. Loss of the *non-redundant* EXCAVATE

holder completes only 5%. Holding the hauler fault fixed and removing the backup hauler flips completion from **100% to 7%**: the same fault, with the outcome decided by whether a capable peer exists. Centralized or decentralized control makes no difference here, since a centralized planner with a single excavator fails identically. The A* layer stays collision-free by construction as obstacle density rises, at $0/1.9/6.5\%$ path-length overhead (a straight-line follower would cut through $0/3.0{\pm}0.5/6.4{\pm}0.8$ obstacles per mission). Makespan grows from $65.0{\pm}3.5$ s at 6 tasks to $119.4{\pm}5.2$ s at 15 (all complete). A fixed-workload fleet-size sweep saturates by six robots ($105.3{\rightarrow}72.6$ s from three to six robots at 200 trials/size, flat thereafter; the apparent rise at eight in earlier 40-trial sweeps does not survive replication).

The fleet-size sweep under physics. *Webots, per-namespace Nav2.* We reran the team-size axis on a superset arena (the deployed world plus two cloned haulers, spawned off the depot column), with the same nine-task workload, excavators pinned at one, and eight reps per size $n \in \{3, 4, 5, 6\}$, interleaved. All runs are clean: zero cap-truncations, zero `physics_artifact` quarantines. Completion saturates: $5/8$ at $n{=}3$ (all three halts at $8/9$ on a single-hauler wedge-down) against $8/8$ for every $n \geq 4$. Completed-mission makespan keeps falling past the kinematic saturation point ($305{\pm}97 \rightarrow 236{\pm}48 \rightarrow 197{\pm}30 \rightarrow 194{\pm}34$ s; adjacent-size Welch $t \leq 1.7$), because extra haulers absorb stall-and-rescue tail events the kinematic layer does not model. Near-misses stay flat at 1–2 per rep, wedge-downs fall to zero by $n{=}5$, and there are zero collisions in all 32 runs. No coordination crossover appears through $n{=}6$, so the abstracted sweep *under*-estimates useful fleet size.

C. Allocator comparison

Kinematic backend. Six allocators drop in behind one interface with no other code change: greedy nearest-compatible, optimal assignment (Hungarian [10]), CBBA [9], and three greedy ablations (random, farthest, round-robin). The comparison runs at 20 reps/cell (nominal and hauler-fault) over the deployed single-excavator fleet, whose capability constraints pin most of the assignment, and a two-excavator fleet that opens it (Table VI). Greedy sits within 0.2% of Hungarian on nominal makespan in both. The two-excavator hauler-fault cell separates them by only $+3.3\%$ (93.4 vs. 90.4 s), inside the 95% CIs ($\pm4.5/\pm4.3$ s), and CBBA tracks the optimum within 3.7% everywhere. The ablations separate: farthest degrades makespan by up to 48%, round-robin 32%, random 23%, and on the capability-pinned fault cell they at least double recovery latency (51–68 s vs. $\sim$26 s). Allocation costs greedy 9–11 μs, Hungarian 11–13 μs and CBBA 287–365 μs per `assign()`, under 0.7 ms cumulative per mission, five orders below makespan.

Webots, per-namespace Nav2. We replayed the two-excavator cell through the full stack with 15 nominal + 15 hauler-fault reps per allocator on the same binary, all 180 trials counted on a verified five-robot fleet. The greedy-vs-optimal makespan ordering does not survive: the kinematic $+3.3\%$ fault-cell deficit inverts to -9% ($256{\pm}19$ vs. $283{\pm}24$ s, CBBA $275{\pm}22$ s, overlapping CIs), so among these three, run-to-run variance in the physics exceeds the allocator effect. The ablation separation survives and turns from delay into failure. Under the fault, round-robin completes $2/15$, farthest $6/15$ and random $10/15$, against $15/15$ for greedy, Hungarian and CBBA: the extra travel and crossings become wedges that a fault-thinned hauler pool cannot absorb.

TABLE VI
ALLOCATOR COMPARISON. KINEMATIC BACKEND: 20 REPS/CELL, NOMINAL / HAULER-FAULT MEAN MAKESPAN OVER COMPLETED MISSIONS; EVERY ALLOCATOR COMPLETES $20/20$ IN EVERY CELL; 95% CIS ±0.8–4.5 S (±3–10 S FOR RANDOM). WEBOTS WITH NAV2: COMPLETIONS AND TASK FRACTION IN THE TWO-EXCAVATOR HAULER-FAULT CELL, 15 REPS. FROM `RESULTS/ALLOCATOR_COMPARISON.MD` AND `RESULTS/ALLOC_PHYSICS.MD`.

	Kinematic makespan (s)		Webots fault cell	
Allocator	1-exc	2-exc	Compl.	Task frac.
Greedy nearest (ours)	77.7 / 92.4	66.3 / 93.4	15/15	1.00
Hungarian	77.8 / 94.8	66.4 / 90.4	15/15	1.00
CBBA	77.1 / 93.3	66.2 / 93.8	15/15	1.00
Random-compatible	95.6 / 109.2	70.0 / 90.6	10/15	$0.90{\pm}0.10$
Farthest-compatible	110.0 / 133.4	73.2 / 133.5	6/15	$0.83{\pm}0.10$
Round-robin	79.5 / 118.8	79.5 / 119.5	2/15	$0.54{\pm}0.13$

D. Full-physics replay: the abstraction gap

Webots physics against the discrete-event simulator. We reran a matched subset (eight no-fault and eight randomized single-hauler-fault reps of the deployed nine-task mission, on the per-namespace Nav2 stack with the §VII-E layer and rescue ladder engaged) through full Webots physics and set the distributions beside abstracted runs replaying each physics trial's fault schedule. These are *residual* gap numbers for the hardened stack. Two fault reps hitting the 600 s cap mid-recovery are excluded from rates, and the auditor quarantined zero runs.

Completion is ground-truth-verified. Wheel odometry keeps integrating while a wedged robot's wheels slip, so NavigateToPose can report success with the true body 1.5–3 m from goal. Completion is therefore gated on the supervisor's ground-truth pose at the 0.45 m Nav2 acceptance radius: a claimed arrival beyond the gate is refused, the robot is downed as wedged, and its task returns to the pool.

Table VII gives the result. Under the abstraction every mission completes by construction. Under physics the hardened fleet matches it to within measurement noise: task fraction 0.99 ± 0.03 with $7/8$ missions complete nominally, and $6/6$ at 1.00 under the hauler fault. The single nominal halt ($8/9$ tasks, one wedged hauler) fell in the noisiest localization window (14 re-seeds against a campaign mean $\approx$3.6), and we attribute it to that noise. Phantom arrivals, $\approx$2 per trial on the pre-Nav2 follower, fall to 0.0. What the abstraction still cannot see is cost. Clean nominal completions take 197 ± 40 s against the matched abstracted 78 s ($+152\%$), and fault recovery widens this to 313 ± 136 s against 84 s ($+273\%$): AMCL/MPPI overhead, conservative local control,

TABLE VII
ABSTRACTION GAP: MATCHED PHYSICS VS. DISCRETE-EVENT RUNS ON THE DEPLOYED FOUR-ROBOT MISSION, WITH GROUND-TRUTH-VERIFIED COMPLETION. THE ABSTRACTION COMPLETES EVERY MISSION BY CONSTRUCTION; PHYSICS DOES NOT.

Scenario	Task compl. (frac.)	Missions compl.	Slip arr. /trial
No fault—physics	0.99 ± 0.03	7/8	0.0
No fault—abstracted	1.00	8/8	0.0
Hauler fault—physics	1.00 ± 0.00	6/6	0.0
Hauler fault—abstracted	1.00	6/6	0.0

escape-and-rescue detours and physical fault handoffs, none of which exist in the discrete-event world. The reality gap [26] here has a direction on each axis: the campaign over-counted completion on the unhardened stack and still under-counts cost on the hardened one.

E. Closing the Σ gap; deep precedence DAGs

Webots, per-namespace Nav2. With the priority-BVC layer of §VII-E enabled and everything else fixed, a rehearsal of the §VIII-A failure goes differently: the hauler yields, curves around the stationary excavator at ≈ 0.9 m clearance, and delivers. The full mission, which previously halted at 6–7/9, **completes** $9/9$ **across three consecutive runs** (our repeatability bar) with every credited arrival ground-truth-gated (all 13 within 0.39 m of goal, mean 0.23 m), minimum inter-agent separation 0.72 m against a 0.36 m contact threshold, zero overlaps, and makespan $\approx$156–168 s.

The same stack drives deeper precedence. The depth-7 linear chain (survey $\rightarrow$ excavate $\rightarrow$ base $\rightarrow$ rebar $\rightarrow$ pour $\rightarrow$ inspect spine with leaf offshoots, 13 tasks) completes $13/13$ at $\approx$265 s, each spine task dispatching only after its predecessor's ground-truth completion. The depth-9 stacked diamond (16 tasks) completes $16/16$, sibling branches overlapping (peak concurrency 2–3) while every sync task waits for its full branch set. Here two-excavator redundancy runs live: an Exc1 fault mid-drive reassigns both digs to the backup, which completes both footings, still $16/16$. One spare machine removes the single point of failure that cost 95% of completions in the abstracted campaign. One navigation change was required: an INSPECT goal inside a fresh pit's inflated keep-out now sends the surveyor to a line-of-sight standoff clear of the inflation, where before its nav aborts on the lethal cell would down the sole INSPECT agent. An eight-rep-per-tier pilot soak validated the §VII-E hardening: chain $3/8 \rightarrow 6/8$ (task-fraction $0.769 \rightarrow 0.971$), diamond $1/8 \rightarrow 3/8$ ($0.445 \rightarrow 0.820$), flat baseline $3/8$ at 0.917, gates clean in all 24/24 runs.

The pilot also traced the dominant deep-tier casualty to *localization*, with pit floors and host load ruled out. Scan-matching in the growing, unmapped pit field drifts the belief half a metre or more off truth, the planner routes on that belief, and the reactive escape drives the stalled agent away from a goal on a pit's far side, until an unrescued sole-capability loss freezes the fleet. The ground-truth goal-drive of §VII-C removes that freeze; one mission finished its full task set through two mid-run downs on redundancy and rescue alone.

F. Final tier soak

Webots, per-namespace Nav2. The final campaign runs four tiers at **eight full-physics reps per tier** on a dedicated 4-core native-Linux host: nine disconnected roots (depth 1), the 13-task chain (depth 7), the 16-task stacked diamond (depth 9), and a 24-task earthworks-and-foundation sequence (depth 15). Every run is validated automatically: precedence gate and sibling-branch parallelism checked from the 10 Hz ground-truth trace, completion ground-truth-gated as in §VIII-D, every halt attributed by the auditor. Sim-time budgets scale with depth (800 s flat/chain, 1400 s diamond, 2000 s full-construction).

Completion is $8/8$, $8/8$, $7/8$ and $0/8$ at mean task-fractions 1.000, 1.000, 0.961 and 0.307, with completed-mission makespans of 167–272 s, 253–730 s and 699–1076 s on the first three tiers. The flat and chain tiers, at $3/8$–$6/8$ in every earlier soak, saturate, and the chain absorbs four mid-run hauler wedges through capability redundancy without losing a mission. The diamond's one halt stops late, at $11/16$, on a capability block. The deepest tier does not complete on this substrate. Localization churn scales with a mission's exposure time to the unmapped pit field (mean re-seeds per run $3.1 \rightarrow 9.6 \rightarrow 13.6 \rightarrow 25.5$ across tiers), and over a 24-task horizon the fleet loses its haulers (11 wedge-downs across eight runs) or its sole excavator (4) to drift-wedges before the spine clears. Escape-command hold ratios stay at 2–3% (74% in the baseline). Precedence gates pass in all 32 runs, parallelism is demonstrated in 31 (the last, a $3/24$ halt, exposes no sibling-overlap window), and every credited task is confirmed by ground truth, so we read the residual as an effect of embodiment and environment.

Two rules govern how these rates are read. Localization noise varies over time enough to confound rates across windows: on a shared host an identical binary swung from $0/3$ to $5/5$ tier-one completions in adjacent $\sim$20-minute windows, tracked by a covariate (AMCL re-seeds per run: completions cluster at 9–17, halts at 20–30). All comparisons here are same-window with the covariate logged, and differences smaller than a few runs in eight are not interpreted. The auditor also quarantines substrate artifacts: a run whose body vertical coordinate leaves the traversable surface ($z < -0.2$ m, an ODE contact-model failure on carved pit slopes) is tagged `physics_artifact`, so we report both an *embodiment* and a *logical* completion rate.

IX. LIMITATIONS

- **Simulation only.** Nothing in this project has run on a physical robot. The fault models are not validated against field failure data, and the fault mix in §V is a sampling choice.
- **Fault models are partly addressed.** The earlier version of this work modeled a fault as a kill signal. The library replaces that with eight typed faults acting on the drive, a tool, the radio link or the site, detected through symptoms. What remains: every fault is a clean on/off model, with no partial power, packet-loss rate, latency,

brownout or load-dependent slowing, and a blockage is an axis-aligned box that appears and clears instantly.

- **Campaign bodies are kinematic.** All 6,000 labeled shifts follow routes exactly, with no slip, inertia or contact. The PhysX comparison covers seven schedules on one site with one seed, and three of the seven changed outcome or task count. Until PhysX campaigns exist, labels that depend on gate traffic hold for the kinematic model only.
- **No sensor model.** Sensing an unmapped obstacle is range plus line of sight, with no lidar or camera model and no noise. Work is a timed dwell: nothing is dug, grasped or placed.
- **One coordinator.** The labels describe AUTON's baseline coordinator and its four presets. It has no lane reservations, so head-on meetings occur without any fault, and the deadlock counts reflect that design. Its watchdogs, traffic rules and deadlock breaker have no Webots or Nav2 validation; only the allocator module, the precedence rule and the grid planner are shared with the stack of §VIII. No third-party fleet software has been connected, and no policy or world model has been trained on the data.
- **Attribution is narrow.** Replay covers deadlocks and duplicate work, tests one alternative for one decision, and runs at most twice per shift.
- **Two sites, small fleets**: five and seven robots, 16 and 24 tasks. The Isaac Sim loop has no ROS 2 or Nav2, no articulated boom or bed, no soil model and no recorded warehouse runs (§VI).
- **The Webots validation keeps its own limits.** The §VIII-B statistics come from an abstracted simulator and are upper bounds (§VIII-D); the team-size axis is physics-validated only to $n{=}6$. Σ is partially encoded, with no approach/exit-path co-design, no closed-form collision or deadlock guarantee under the priority asymmetry, and static hand-assigned priorities (a learned weighting [14] is unaddressed). The greedy allocator stays within CIs of Hungarian and CBBA ($\leq 3.3\%$ kinematic, -9% under physics) on one arena, one fault class and a nine-task mission. Deep-tier completion is bound by localization noise in the growing pit field (§VIII-F).

X. Conclusion

AUTON takes a site file, runs simulated shifts on it, injects eight typed faults that the coordinator can detect only through symptoms, and returns one label per shift stating what failed, which numbered decision detected and answered it, and whether the fleet recovered. Where a coordinator decision sits at the root of a deadlock or of duplicate work, the label names it and reports whether a replay with that decision changed avoided the incident. On kinematic bodies, 6,000 shifts across two sites produced 8,998 labeled faults and 330,915 logged decisions, and 84.2% of 4,366 one-decision replays removed the incident they targeted. On PhysX in Isaac Sim, four of seven schedules matched their kinematic twins and three diverged, two of them in gate traffic. The allocator, precedence rule and planner under test were measured earlier in Webots under ROS 2/Nav2: task fraction $0.99{\pm}0.03$ against the abstraction, a residual 2.5–$3.7\times$ makespan inflation, and precedence gates clean in all 32 soak runs. Every result here is simulated. The next steps follow from the limitations: PhysX campaigns at the scale of the kinematic one, a second coordinator behind the policy interface, fault parameters fitted to field data, and a model trained on the labels.

References

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[3] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[4] H. Ma and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proc. AAMAS*, 2017.

[5] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," in *Proc. AAAI*, 2021.

[6] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Trans. Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[7] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[8] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[9] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[10] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.

[11] L. E. Parker, "ALLIANCE: An architecture for fault-tolerant multirobot cooperation," *IEEE Trans. Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[12] X. Zhou et al., "Fast, on-line collision avoidance for dynamic vehicles using buffered Voronoi cells," *IEEE RA-L*, vol. 2, no. 2, pp. 1047–1054, 2017.

[13] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. ECC*, 2019, pp. 3420–3431.

[14] A. Pierson, W. Schwarting, S. Karaman, and D. Rus, "Weighted buffered Voronoi cells for distributed semi-cooperative behavior," in *Proc. ICRA*, 2020, pp. 5611–5617.

[15] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research (ISRR 2009)*, Springer Tracts in Advanced Robotics, vol. 70, 2011, pp. 3–19.

[16] R. L. Johns et al., "A framework for robotic excavation and dry stone construction using on-site materials," *Science Robotics*, vol. 8, no. 84, eabp9758, 2023.

[17] D. Jud et al., "HEAP—The autonomous walking excavator," *Automation in Construction*, vol. 129, 103783, 2021.

[18] H. A. D. Nguyen and Q. P. Ha, "Robotic autonomous systems for earthmoving equipment operating in volatile conditions and teaming capacity: a survey," *Robotica*, vol. 41, no. 2, pp. 486–510, 2023.

[19] X. Ye, H. Guo, and Z. Luo, "Two-stage task allocation for multiple construction robots using an improved genetic algorithm," *Automation in Construction*, vol. 165, 105583, 2024.

[20] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[21] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *Int. J. Robotics Research*, vol. 27, no. 3–4, pp. 463–479, 2008.

[22] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An autonomous robotic system for three-dimensional collective construction," in *Proc. Robotics: Science and Systems (RSS)*, 2011.

[23] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[24] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A navigation system," in *Proc. IROS*, 2020.

[25] O. Michel, "Cyberbotics Ltd. Webots: Professional mobile robot simulation," *Int. J. Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[26] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *ECAL 1995*, LNCS vol. 929, Springer, 1995, pp. 704–720.

[27] K. Perlin, "Improving noise," *ACM Trans. Graphics (SIGGRAPH)*, vol. 21, no. 3, pp. 681–682, 2002.

[28] C. Boje, A. Guerriero, S. Kubicki, and Y. Rezgui, "Towards a semantic construction digital twin: Directions for future research," *Automation in Construction*, vol. 114, 103179, 2020.
[29] Y. Xu et al., "Robotics in the construction industry: A bibliometric review of recent trends and technological evolution," *Applied Sciences*, vol. 15, no. 11, 6277, 2025.
[30] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. SSC*, vol. 4, no. 2, pp. 100–107, 1968.
[31] NVIDIA, "Isaac Sim documentation," software documentation. [Online]. Available: https://docs.isaacsim.omniverse.nvidia.com/
[32] J. Carlson and R. R. Murphy, "How UGVs physically fail in the field," *IEEE Trans. Robotics*, vol. 21, no. 3, pp. 423–437, 2005.
[33] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Computing Surveys*, vol. 51, no. 1, art. 9, 2018.
[34] A. Basiri et al., "Chaos engineering," *IEEE Software*, vol. 33, no. 3, pp. 35–41, 2016.
[35] D. Ha and J. Schmidhuber, "World models," arXiv:1803.10122, 2018.