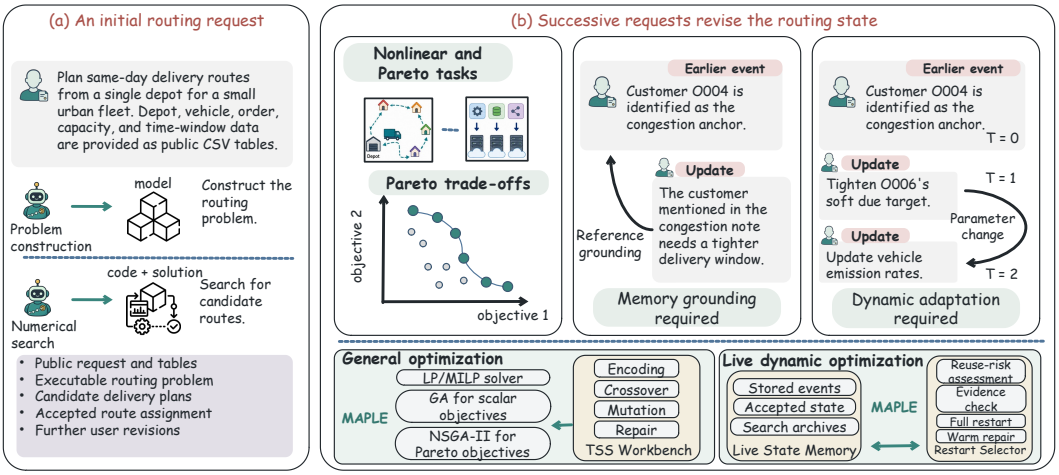
MAPLE: MEMORY-AUGMENTED PLANNING WITH LANGUAGE AND EVOLUTION

Research code **Harness application** **Online demo**

Anonymous authors
Paper under double-blind review

ABSTRACT

Operational plans often need to be revised as requirements change over time. Responding to these changes requires updating the underlying optimization problem, preserving decisions that users wish to keep, and deciding whether to reuse previous solutions as starting points or search from scratch. We introduce MAPLE (Memory-Augmented Planning with Language and Evolution), an agent for optimization under changing requirements expressed in natural language. MAPLE constructs the initial optimization problem through a typed program interface and solves it with numerical solvers. For each subsequent request, it revises and solves the current problem, uses records of prior requests and a saved plan to identify and preserve decisions that users ask to keep, and selectively reuses previous solutions to initialize the new search. To evaluate agents under changing requirements, we introduce NLDO, a benchmark of 15 planning tasks, each followed by 12 natural-language updates, spanning five task types: resource selection and allocation, production scheduling, staff scheduling, delivery route planning, and computing task allocation to servers. In the main evaluation, MAPLE completes all 15 task sequences, achieving mean online single-objective quality of 0.951 and a mean Pareto hypervolume ratio of 0.875 across initial and updated task states.



Native capabilities and evaluated adaptations

Method	Natural language	Solver output	Retained Pareto set	Live updates	Solver state reuse
ReAct	Yes	Yes	No	Part.*	No
OR-LLM-Agent	Yes	Yes	No	Part.*	No
ORLM	Yes	Yes	No	Part.*	No
OptiMUS	Yes	Yes	No	No	No
OptimAI	Yes	Yes	No	No	No
Persistent ReAct	Yes	Yes	No	Yes*	Part.*
MAPLE (ours)	Yes	Yes	Yes	Yes	Yes

Primary results by static/dynamic setting

Method	NLP4LP-hard	BWOR20	NLDO SS	NLDO DS	NLDO SM	NLDO DM
ReAct	57.6%	70.0%	0.889*	0.333*	0.300*	0.023*
OR-LLM-Agent	50.8%	65.0%	0.778*	0.111*	0.440*	0.046*
ORLM	23.7%	45.0%	**1.000***	0.120*	0.000*	0.000*
OptiMUS	**69.5%**	65.0%	0.889*	0.281*	0.417*	0.084*
OptimAI	61.0%	55.0%	0.556*	0.077*	0.377*	0.051*
Persistent ReAct	62.7%	70.0%	0.889*	0.501*	0.259*	0.042*
MAPLE (ours)	59.3%	**80.0%**	**1.000**	**0.951**	**0.832**	**0.875**

Figure 1: **MAPLE maintains plans through successive requests.** The routing example shows problem revision, memory of earlier decisions, and reuse of search results. The tables compare capabilities and performance. An asterisk marks adapted baselines.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Production planners, dispatchers, and workforce coordinators understand operational requirements but may lack the specialist support needed to formulate optimization models and maintain solver programs. This expertise barrier limits everyday adoption of optimization methods (AhmadiTeshnizi et al., 2024; Zhang et al., 2025). LLM-based agents translate natural-language descriptions into executable models or solver code and use execution feedback to repair errors (AhmadiTeshnizi et al., 2024; Huang et al., 2025; Thind et al., 2025; Zhang et al., 2025). Recent systems also clarify planning requirements with users and construct models from business data (Xie, 2026).

Operational planning, however, continues after the first solution. New orders arrive, resources become unavailable, and users revise constraints or priorities. Consider a dispatcher who first records a congested customer and later asks to keep that customer's vehicle assignment. The agent must identify the customer from the earlier request, retrieve its accepted assignment, and enforce that assignment in the revised problem. Separately, it must decide whether earlier candidate solutions remain useful for the next search. Preserving a required decision and reusing a promising candidate are different tasks: the former constrains the new solution, whereas the latter changes how the solver searches for it.

Related work addresses parts of this setting. Multi-turn planning benchmarks study revisions as new constraints arrive (Oh et al., 2025); LLM-based optimization methods address nonlinear problems and multi-objective search (Thind et al., 2025; Schwanke et al., 2026); dynamic optimization studies population repair, transfer, and restart under changing objectives and constraints (Branke, 2002; Nguyen et al., 2012). Our focus is their intersection: maintaining an executable optimization problem through cumulative natural-language revisions, while recovering historical decisions and carrying useful numerical search state. We call this setting *live dynamic optimization*. The setting includes both nonlinear and multi-objective tasks.

MAPLE maintains a generated optimization program, called the *Workbench*, together with its data, event history, representative plan, solution archive, and search population. At initialization, the model specializes a runnable template through a typed interface; fixed code supplies the corresponding operators and solvers. Each later request first updates the relevant data or functions and resolves any historical assignment requirements. For evolutionary search, a separate selector then chooses between repairing part of the previous population and initializing a fresh population. Program and output checks control commitment, while established solvers perform numerical optimization.

We evaluate MAPLE on NLDO: 15 trajectories and 180 updates across five domains, measuring continuity and plan quality. In the main evaluation, MAPLE completes all trajectories, reaching online scalar quality of 0.951 and a Pareto hypervolume ratio of 0.875. The experiments distinguish output-protocol failures from mathematical infeasibility, and separately test program construction, historical assignment retrieval, and search initialization.

Our contributions are threefold. First, we formulate optimization under cumulative natural-language revisions and introduce NLDO, with static and dynamic views of scalar and multi-objective tasks. Second, we develop MAPLE, which integrates executable problem revision, historical decision retrieval, and numerical-state reuse. Third, controlled comparisons identify how typed construction, complementary history records, and selective reuse affect continuity and quality.

2 TASK AND BENCHMARK

Optimization under successive requests. A trajectory starts with a natural-language request and input tables (x_0, D_0), followed by revisions $u_{1:T}$. Here, *public* inputs are those available to the agent. The initial request and cumulative revisions define a feasible set $\mathcal{F}_t$ and objective vector $\mathbf{f}_t = (f_{t,1}, \ldots, f_{t,k_t})$ at state t. For $k_t = 1$, the agent should return a feasible plan with a small objective value; for $k_t > 1$, it should return feasible, mutually nondominated alternatives that represent the trade-offs. Objectives are converted to minimization for evaluation. Requests distinguish optimization objectives, hard constraints, and auxiliary diagnostics.

Every method receives the initial request, public tables, cumulative updates, output requirements, and execution feedback, and maintains its own history. Section 4 and Appendix A.8 specify the state exposed to each evaluated workflow. Section 4 defines the separate continuity and quality measures.

NLDO coverage. NLDO contains five domain families with three trajectories each. Every trajectory has one initial state and twelve updates, giving 15 trajectories, 195 states, and 180 updates. Table 1 maps the scalar and Pareto domains to the four evaluation views. Dynamic views include initialization (t00–t12); update-only analyses use t01–t12. Numerical-search seeds repeat optimization within a fixed language-model run.

Domains and trajectories	Initial state: t00	Full trajectory: t00–t12
Scalar: selection, scheduling, rostering (9)	NLDO-SS	NLDO-DS
Pareto: routing, cloud placement (6)	NLDO-SM	NLDO-DM

Table 1: NLDO views. Each of the five domains contributes three trajectories.

The main trajectories retain their initial table schemas. Rostering, routing, and cloud placement include disruptive replacements at t11–t12. Other updates change feasibility, reorder candidates, rescale objectives, or change only diagnostics; a changed input therefore need not change the Pareto ordering. Appendix A.6 details these distinctions. Separate controls test historical assignments and reuse under renaming or task addition (Appendix A.13).

3 METHOD

MAPLE first constructs an executable problem and then revises it as requests arrive. Its committed state is

$$S_t = (D_t, W_t, A_t, P_t, \hat{y}_t),$$

where D_t contains input tables, W_t is the Workbench, A_t is the returned solution archive, P_t is the evolutionary population, and $\hat{y}_t$ is the saved *online representative* used for historical assignments. The event history E_t stores accepted requests and their resolved updates. An accepted revision commits the new state and appends its event; a rejected revision leaves the preceding state and history unchanged.

Three components implement this workflow. Typed Search-Space Scaffolding (TSS) supplies a restricted program interface and type-specific operators. Live Problem Decomposition (LPD) localizes the requested data and function edits. Live State Memory (LSM) retains the records needed to resolve historical references. A separate restart selector decides how to initialize the next evolutionary search.

TSS: specialize the program and bind typed operators. The initialization prompt supplies a complete runnable Workbench template. The model specializes two functions. `build_problem(public_context)` reads the input tables, constructs evaluation data, and declares the solver, objectives, and decision segments. A segment contains binary, categorical, bounded integer, bounded real, permutation, or assignment decisions. Their composition defines an encoded candidate, or *genome*. `evaluate(genome, data)` decodes that candidate into a plan and returns objectives and constraint violations. The fixed runtime checks and executes this problem-specific program.

Each declared type selects fixed initialization, crossover, mutation, and domain-repair operators. Real vectors use simulated binary crossover and polynomial mutation (Deb et al., 2002); permutations use order crossover and swap mutation (Davis, 1985); discrete vectors and assignments use gene-wise parent choice and type-valid mutation (Holland, 1975; Syswerda, 1989). These operators preserve the representation domain $\mathcal{X}(W_t, D_t)$. Problem feasibility is narrower: the intended set $\mathcal{F}_t$ also requires the business constraints, which the generated evaluator must implement correctly.

The solver follows the declared formulation. A fixed LP/MILP interface builds a model from variables, bounds, objective coefficients, and constraints (Dantzig, 1963; Nemhauser & Wolsey, 1988). Scalar nonlinear or combinatorial tasks use a genetic algorithm (Holland, 1975), and multi-objective tasks use NSGA-II (Deb et al., 2002). The runtime owns selection, variation, population updates, archive maintenance, and stopping.

Routing example (Figure 2). The data reader retains active orders and available vehicles. An assignment segment a_i selects a vehicle for each order; a permutation π orders the visits. Their typed operators preserve allowed vehicle choices and one occurrence of each active order. The evaluator groups orders by vehicle in permutation order, forming one depot-to-depot route per used vehicle.

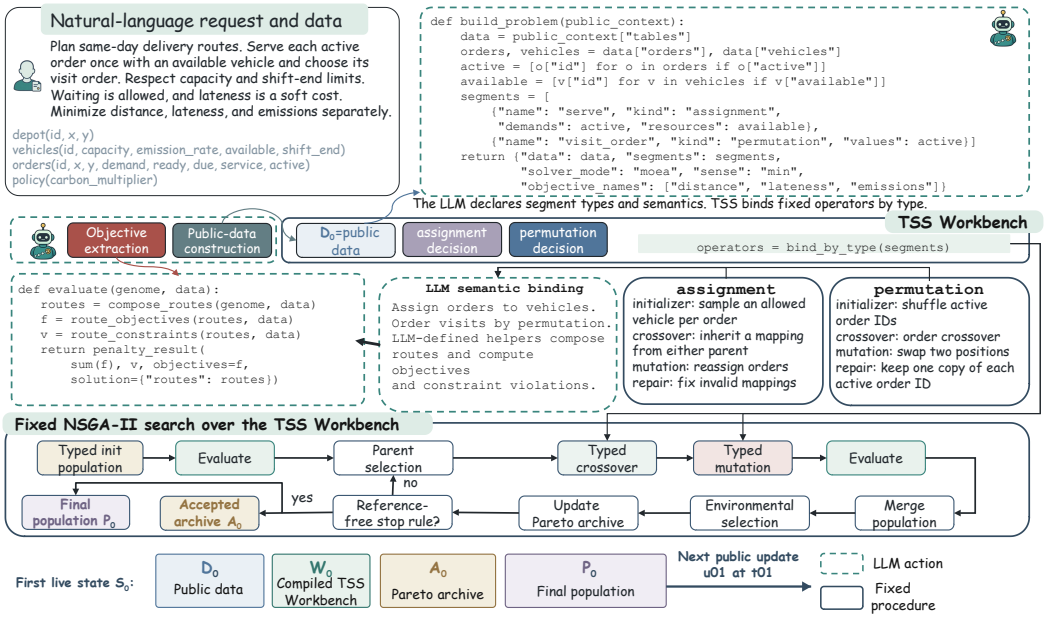


Figure 2: **Constructing the initial state.** The model specializes a runnable Workbench template using D_0. Fixed typed operators and NSGA-II produce population P_0 and archive A_0; the representative $\hat{y}_0$ is stored separately. Dashed teal marks model-authored code; solid blue-gray marks fixed execution. Stopping follows internal search progress.

It computes distance, service-start lateness, and emissions, together with violations for capacity, shift-end return, and missing orders. NSGA-II uses the objective vector. Local repair and ranking use the scalar fitness shown in the diagram.

LPD: locate the requested revision. LPD decomposes each update into data edits, function edits, and historical binding queries. LPD reads the request, current decision declarations, event history, and representative plan. It returns a data-update flag, an edit flag for each Workbench function, historical binding queries, and a provisional restart hint h_t. Supplied table edits are applied directly; a separate model call translates language-only data changes into table edits. The editor receives the flagged functions and this hint, and changes only those functions. A price change can leave both functions unchanged. LP/MILP formulation changes belong in `build_problem()`; evolutionary objectives, constraints, and decoding belong in `evaluate()`. The final restart action is selected after compilation and smoke tests.

LSM: recover and enforce historical assignments. Event history and the representative plan provide different information. For "keep the vehicle of the customer mentioned in the congestion note," LPD emits a typed assignment query. Fixed code resolves the customer from E_{t-1}, reads its vehicle from $\hat{y}_{t-1}$, checks the assignment against the revised decision domain, and checks it whenever a search candidate is evaluated, including after variation. A binding-only request need not change the Workbench functions. The event is added to history only after commitment.

In the recorded NSGA-II runs, $\hat{y}_t$ is the first candidate returned by final population selection. Appendix A.13 distinguishes this online representative from the archive-derived representative used in a separate language test.

Restart selector: choose the next population initialization. After compilation and smoke tests, fixed code compares the old and revised compiled Workbenches and their public tables to construct the public-change summary Δ_t. A separate semantic-selector call uses this summary and accepted request history to assess the usefulness of repaired historical candidates. The fixed gate returns Full only when the response reports high reuse risk, recommends Full, names a structurally supported risk category, and cites a changed field; otherwise it returns Warm. Category support and cited paths are checked separately; the logged explanation does not enter the gate (Algorithm 3). The gated action α_t replaces h_t and determines population initialization.

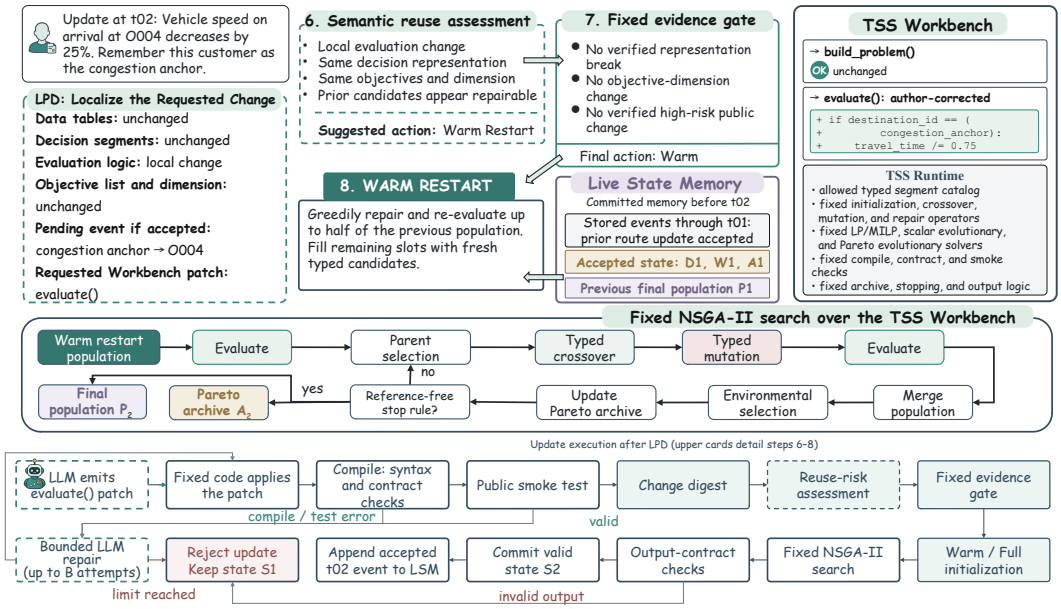


Figure 3: **Revising a routing problem.** The example applies the public travel-time rule and records O004 for later references. LPD locates edits and LSM resolves historical assignments. Program checks precede restart selection, search, and state commitment.

Warm migrates candidates from P_{t-1} and A_{t-1} to the revised decision domains, then applies local typed repair, prioritizing feasibility and lower penalized scalar fitness. Repaired candidates fill at most half the population; fresh candidates fill the rest. Full samples the entire population afresh. Both initialization actions feed the same evolutionary loop and stopping rule. NSGA-II compares objective vectors separately. Warm additionally evaluates candidates during migration and local repair.

Program checks, output checks, and commitment. Before restart assessment, the runtime compiles the edited functions, checks their interfaces, and executes a small public-input example. Failures trigger at most B additional model attempts at *program repair*. After these checks and, for evolutionary search, restart selection, the fixed solver runs and checks the required output fields and structure. A failure rejects the revision; a passing result commits the data, Workbench, archive, population, and representative, and appends the event. These executable and protocol checks are separate from the held-out mathematical feasibility and quality checks used in evaluation. Appendix A.1 gives the full control flow; Appendix A.3 details the program interface and candidate-repair procedure.

4 Experiments

We ask five questions: can an agent solve the initial request, maintain continuity, revise its program, enforce historical assignments, and reuse useful candidates?

Compared agents and repeated runs. We compare MAPLE with ReAct, Persistent ReAct, ORLM, OptiMUS, OR-LLM-Agent, and OptimAI. NLDO adapters supply cumulative public inputs and each method's own history, with at most three code-output repairs. Persistent ReAct is our stateful ReAct control: it carries solver code and a compact answer between requests and can return a Pareto archive within a request, but has no separately carried search population. Appendix A.8 specifies all evaluated interfaces.

The main backbone is DeepSeek-V4-Pro (Preview), API ID `deepseek-v4-pro`. Each MAPLE trajectory uses one language-model run to produce its program revisions and restart choices, followed by ten numerical-search seeds under those fixed outputs. External methods contribute one language-model run per trajectory. Each evolutionary search uses 200 candidates and at most 200 generations; program repair has limit $B = 3$. Numerical error bars therefore measure search variability, not independent agent runs. Separate controls vary program generation and the backbone (Section 4.5).

Evaluation. Search uses the Workbench evaluator. Independent evaluation computes objectives from submitted plans using the public formulas. Each Pareto state has a shared reference formed by pooling ten independent searches with 500 candidates and 500 generations, then retaining the feasible nondominated set. These references are used only for offline scoring. The main agent comparison, complete TSS ablation, full-trajectory restart policies, and Kimi run use the common public-formula evaluation. Single-update restart, operator, independent-program, binding-replay, and reference-sensitivity controls use their recorded evaluators and specified reference sets (Appendix A.6).

Continuity and plan quality. Let a_t be the recorded acceptance indicator, including the original output-protocol and validity checks. Let v_t indicate that the saved output contains at least one plan feasible under the evaluation formulas, and let q_t be its quality, set to zero when $v_t = 0$. Strict-prefix eligibility is $r_t = \prod_{j=0}^{t} a_j$. We report

$$\text{SolveRate} = \frac{1}{T+1}\sum_{t=0}^{T} r_t v_t, \qquad \text{OnlineQuality} = \frac{1}{T+1}\sum_{t=0}^{T} r_t q_t.$$

For scalar tasks, q_t is normalized scalar quality; for Pareto tasks it is the *HV ratio*, the feasible archive's hypervolume divided by that of its reference. The ratio can exceed one because references are finite. IGD uses the same normalization and only eligible nonempty feasible archives.

The mathematical-prefix diagnostic replaces r_t by $\tilde{r}_t = \prod_{j=0}^{t} v_j$, disregarding auxiliary output fields. Both protocols include t00 and give missing states zero. Strict-prefix termination is a scoring rule: some saved external workflows continued executing after an auxiliary-field failure. Runtime commitment, mathematical feasibility, and prefix eligibility are therefore distinct. Appendix A.5 defines the metrics and aggregation in full.

4.1 Q1: Can the agent solve the initial request?

We first test the initial request without later revisions. Table 2 separates successful execution from agreement with the reference objective. MAPLE reaches objective-agreement rates of 59.3% on NLP4LP-hard and 80.0% on BWOR20; OptiMUS reaches 69.5% on NLP4LP-hard. On NLDO initialization, its scalar quality is 1.000 and Pareto HV ratio is 0.832. Appendix A.7 gives tolerances and workflow diagnostics.

	NLP4LP-hard		BWOR20		NLDO-SS		NLDO-SM	
Method	**Exec.**	**Obj.**	**Exec.**	**Obj.**	**Valid**	**Quality**	**Valid**	**HV ratio**
MAPLE	93.2%	59.3%	**100.0%**	**80.0%**	**100.0%**	**1.000**	**100.0%**	**0.832**
Persistent ReAct	98.3%	62.7%	**100.0%**	70.0%	88.9%*	0.889*	50.0%*	0.259*
ReAct	78.0%	57.6%	95.0%	70.0%	88.9%*	0.889*	33.3%*	0.300*
OR-LLM-Agent	96.6%	50.8%	95.0%	65.0%	77.8%*	0.778*	66.7%*	0.440*
ORLM	61.0%	23.7%	55.0%	45.0%	100.0%*	1.000*	0.0%*	0.000*
OptiMUS	93.2%	**69.5%**	**100.0%**	65.0%	88.9%*	0.889*	66.7%*	0.417*
OptimAI	**98.3%**	61.0%	80.0%	55.0%	55.6%*	0.556*	66.7%*	0.377*

Table 2: **Static execution and solution quality.** Exec.: successful execution. Obj.: reference-objective agreement over all cases. Valid: recorded acceptance and offline feasibility both hold. HV ratio uses public formulas and is zero for ineligible or infeasible outputs. Appendix A.7 reports supplementary diagnostic checks. * marks task-interface adaptation.

4.2 Q2: Can the agent follow a sequence of revisions?

We next test continuity across revisions. Recorded acceptance includes the original protocol and validity checks, whereas offline feasibility is recomputed from saved plans under the evaluation formulas. The strict and mathematical prefixes expose the effect of this distinction. Figure 4 shows per-state quality, and Table 3 summarizes continuity and quality over the same sequences.

MAPLE completes every trajectory in the main language-model run, with strict-prefix scalar quality of 0.951 and Pareto HV ratio of 0.875. Persistent ReAct obtains 0.501 and 0.042. Seven of its scalar

Method	Dynamic single-objective			Dynamic multi-objective		
	Solve rate	Strict Quality	Math. Quality	Solve rate	Strict HV ratio	Math. HV ratio
MAPLE	100%	0.951	0.951	100%	0.875	0.875
Persistent ReAct*	51.3%	0.501	0.763	11.5%	0.042	0.185
ReAct*	33.3%	0.333	0.376	2.6%	0.023	0.091
OptiMUS*	28.2%	0.281	0.333	12.8%	0.084	0.118
ORLM*	12.0%	0.120	0.298	0%	0.000	0.021
OptimAI*	7.7%	0.077	0.103	11.5%	0.051	0.059
OR-LLM-Agent*	11.1%	0.111	0.171	12.8%	0.046	0.046

Table 3: **Continuity and quality**, t00–t12. Solve rate uses strict prefixes. Math.: mathematical prefix. Pareto scores use public formulas. Both protocols assign zero to missing states; strict prefixes also exclude saved states after the first recorded rejection. * marks dynamic adaptation.

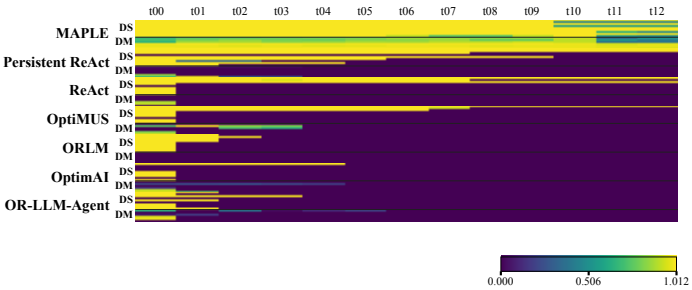


Figure 4: **Strict-prefix quality through updates.** Upper: scalar; lower: Pareto. Columns cover t00–t12. Pareto quality uses public-formula evaluation. A protocol rejection ends the eligible prefix.

trajectories first fail the protocol because a required reasoning field is missing; one of these outputs is also infeasible. Using mathematical prefixes raises Persistent ReAct's scores to 0.763 and 0.185, while MAPLE's scores are unchanged. Thus auxiliary-field failures explain part, but not all, of the measured continuity gap. Table 19 reports both solve rates for every external method.

4.3 Q3: What helps construct and revise the program?

Typed construction and restricted editing. The no-TSS control generates its representation and all initialization, evaluation, variation, and repair callbacks from scratch. It removes typed representation, type-specific operators, and the restricted editing interface together; the numerical loop remains fixed. Across Pareto updates, feasibility falls from 100.0% to 80.6% and mean HV ratio from 0.879 to 0.665 (Table 4). The value 0.879 excludes t00, whereas the main dynamic value 0.875 includes it.

Method	Feasible	HV ratio ↑	IGD
MAPLE	100.0%	0.879±0.014	0.079
Always Warm	100.0%	0.868±0.016	0.091
Always Full	100.0%	0.824±0.015	0.111
w/o TSS	80.6%	0.665±0.004	0.143

Table 4: **Scaffolding and full-trajectory reuse.** Six NLDO-DM trajectories, t01–t12; one language-model run per trajectory and ten numerical seeds. Public-formula HV ratio: mean ± sample SD across seed means, with infeasible states scored zero. IGD uses eligible feasible archives.

The version without TSS returns 140 invalid outputs among 720 update–seed results. Its routing program omits the check that a vehicle serves only one route. Its cloud-placement program fails to reject GPU-incompatible assignments.

Among its feasible outputs, the mean HV ratio is 0.826.

Search operations tailored to the decisions. To separate operator choice from the compound TSS intervention, we change only crossover and mutation. The control copies a complete decision block from one parent or samples a new type-valid block. Workbench, candidate repair, source population, and restart action remain fixed. Across twelve updates of one routing and one cloud-placement trajectory with three numerical seeds, mean HV ratio falls from 0.774 to 0.618 under the recorded evaluator (Table 24). All 13 no-TSS Workbench versions compile, while the missing checks affect subsequent updates (Table 23). The operator control separately links typed variation to search quality.

Update localization and executed edits. LPD's three edit flags agree with the saved data and function modifications in 170 of 180 updates. Both Workbench functions remain unchanged in 166 updates, so only 14 updates modify the generated program. This measures localization consistency with executed edits. The ten disagreements comprise six supplied data replacements and four requested code edits that returned unchanged functions (Appendix A.13).

4.4 Q4: Which history is needed to preserve assignments?

Historical entity lookup. Thirty benchmark requests identify an entity through earlier updates rather than naming its ID. LSM resolves all 30, including six chains with three reference links (Appendix A.13). This tests event-history lookup; the following controls additionally require assignment retrieval and enforcement.

Assignment interpretation. The language test has 18 requests: two domains, three information requirements, and three equivalent wordings. Six need event history only; six name entities and preserve assignments from a representative plan; six need both sources. Full LSM resolves all 18. The events-only and plan-only conditions each resolve their six directly answerable requests, but neither resolves those needing both (Table 22a). These tests use *reconstructed representatives*, selected by minimum recorded scalar fitness from saved archives.

Assignment enforcement. A separate search replay applies the saved first-wording queries to six cases with three numerical seeds each. Assignments come from the original *online representatives*; Workbench, source population, Warm initialization, and search cap are fixed across conditions. Full LSM and the supplied-assignment control succeed in all 18 case–seed results (Table 22b). Success requires correct retrieval and enforcement, with every returned plan feasible and respecting the assignment. Appendix A.13 details the two representative sources and their differences. Events-only and plan-only controls each succeed in six of 18 results, with recorded-evaluator mean HV of 0.261 and 0.258 versus 0.825 for full LSM. History identifies the entity, while the plan supplies its earlier assignment.

4.5 Q5: When should search reuse previous candidates?

Full-trajectory policies. Always Warm and Always Full fix the initialization action at every update, and each policy carries the population from its own preceding searches. These policies test cumulative reuse, unlike *Matched Warm/Full*, which test one update from MAPLE's same saved population (Appendix A.13). Over six Pareto trajectories, mean update HV ratio is 0.879 for MAPLE, 0.868 for Always Warm, and 0.824 for Always Full under public-formula scoring.

MAPLE and Always Warm both obtain 0.915 on t01–t10. At t11–t12, substantial task and resource replacements lead MAPLE to select Full; it obtains 0.696 versus 0.636 for Always Warm, with gains concentrated in routing (Figure 5). Across the six trajectory-level paired differences, the mean gain is 0.010, with a trajectory-bootstrap 95% interval of [0.001, 0.021]. Five differences are positive (two-sided exact sign test, $p = 0.219$).

Large input changes without losing useful candidates. We next test four renaming or low-demand task-addition cases, each from the same saved source population with three numerical seeds. Two comparison rules choose Full based on either the fraction of changed table cells or re-evaluation of old candidates. The table-change rule matches MAPLE on the main Pareto trajectories; these additional cases separate changed inputs from lost candidate utility.

MAPLE selects Warm in all four added cases, whereas the table-change rule selects Full. With task additions, Warm exceeds Full by 0.708 HV ratio in routing and 0.022 in cloud placement, using recorded evaluators and separate pooled references (Table 5). In the renaming tests, the numerical rule initially evaluates old identifiers without applying the available mapping and selects Full. Mapping the identifiers first changes all six seed-level decisions to Warm (two domains, three seeds). Thus numerical re-evaluation recognizes reuse after representation alignment.

Uniformly scaling the routing objectives by ten preserves feasibility and Pareto dominance. Warm exceeds Full by 0.325 ± 0.299 in HV ratio (paired mean $\pm$ SD, three seeds, control-specific pooled reference, Table 31). Large objective changes can therefore leave prior candidates useful.

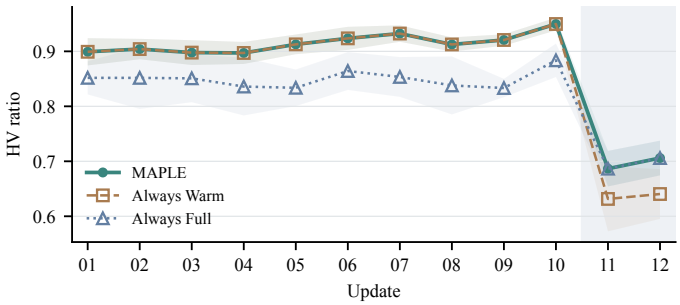


Figure 5: **Reuse before disruption, restart after it.** Public-formula HV ratio over six trajectories: ten-numerical-seed means and one-SD bands under fixed model outputs. Each policy carries its own population; gray marks disruptive updates.

Change	Routing	Cloud
Renamed entities	+0.149	0.000
Added tasks	+0.708	+0.022

MAPLE action: Warm in all four cases.

Table 5: **Reuse after renaming or task addition.** Warm minus Full HV ratio; three paired numerical seeds, recorded evaluators, and case-specific pooled references.

Robustness and cost. Independent-program controls separate model variability from numerical seeds (Table 25); MAPLE's two additional cloud program sequences each solve 83.3% of states. A Kimi k2.7 run obtains scalar solve rate/quality of 100.0%/0.937 and Pareto solve rate/HV ratio of 92.3%/0.766 (Table 26). Code construction uses 51.6k tokens per MAPLE trajectory versus 72.8k without TSS across four additional runs per method; the latter reuses localization and data patches, so this comparison excludes other agent costs. Generation counts track the evolutionary loop. Stage wall time also includes migration and repair. Appendices A.3 and A.13 give overheads and full controls.

5 Related Work

Natural-language optimization. NL4Opt maps language to linear programs (Ramamonjison et al., 2023). Agents generate formulations and solver code with execution feedback (AhmadiTeshnizi et al., 2024; Zhang et al., 2025; Huang et al., 2025; Thind et al., 2025); MIRROR and ORPilot add requirement elicitation and business-data handling (Shi et al., 2026; Xie, 2026). OR-Space studies optimization-model revision in persistent workspaces (Zhou et al., 2026). MAPLE focuses on revising the executable problem while retaining decisions and search state.

Interactive agents and memory. Agents retain dialogue and experience (Yao et al., 2023; Packer et al., 2023; Maharana et al., 2024). Flex-TravelPlanner introduces constraints over multiple turns (Oh et al., 2025); Gaia2 tests asynchronous events and environment-changing actions (Froger et al., 2026). AdaPlanBench studies replanning under evolving constraints (Liu et al., 2026). MAPLE distinguishes event history for entity lookup from representative plans for assignment retrieval; NLDO measures both continuity and optimization quality.

Dynamic and multi-objective optimization. Population repair, transfer, and restart address changing objectives and constraints (Branke, 2002; Nguyen et al., 2012). MOHOLLM uses LLM surrogates and samplers for multi-objective search (Schwanke et al., 2026). MEoH and E2OC evolve heuristics or operators (Yao et al., 2025; Qiu et al., 2026). MAPLE instead uses language for problem maintenance and initialization selection, with numerical search performed by established solvers such as NSGA-II (Deb et al., 2002).

6 Conclusion

MAPLE maintains executable optimization problems across natural-language revisions, separating program edits, historical assignment retrieval, and candidate reuse. In the main evaluation it completes all 15 NLDO trajectories, with online scalar quality of 0.951 and Pareto HV ratio of 0.875. Controls link typed construction to fewer infeasible outputs, show when assignments require both event history and a representative plan, and locate restart gains in disruptive routing updates. Repaired candidates remain useful after renaming or task addition. Together, the results support maintaining problem state while evaluating protocol continuity and mathematical quality separately.

Ethics and reproducibility. NLDO uses synthetic data. The supplement provides tasks, prompts, settings, seed-level evidence, and offline scoring with references separate from agent inputs (Appendix A.10).

AI USE STATEMENT

LLMs assisted with implementation, debugging, manuscript organization, and language editing. DeepSeek-V4-Pro (Preview) and Kimi k2.7 served as the evaluated model backbones described in the experimental settings. The authors are responsible for the final results, claims, and artifacts.

REFERENCES

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. *arXiv preprint arXiv:2402.10172*, 2024.

J. E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. doi: 10.1057/jors.1990.166.

Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011. doi: 10.1016/j.trb.2011.02.004.

Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012. doi: 10.1016/j.future.2011.04.017.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41:157–183, 1993. doi: 10.1007/BF02023073.

Jürgen Branke. *Evolutionary Optimization in Dynamic Environments*. Springer, 2002.

Sara Ceschia, Nguyen Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. The second international nurse rostering competition. *Annals of Operations Research*, 274:171–186, 2018. doi: 10.1007/s10479-018-2816-0.

George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

Lawrence Davis. Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pp. 162–164, 1985.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Kalyanmoy Deb, N. Udaya Bhaskara Rao, and S. Karthik. Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pp. 803–817, 2007. doi: 10.1007/978-3-540-70928-2_60.

Sevgi Erdoğan and Elise Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, 2012.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9gw03JpKK4.

John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 2025. arXiv:2405.17743.

Jiayu Liu, Cheng Qian, Zhenhailong Wang, Bingxuan Li, Jiateng Liu, Qing Zong, Heng Wang, Jeonghwan Kim, Yumeng Wang, Bingxiang He, Xiusi Chen, Yi R. Fung, and Heng Ji. AdaPlanBench: Evaluating adaptive planning in large language model agents under world and user constraints, 2026. URL https://arxiv.org/abs/2606.05622.

Min Liu, Jinhua Zheng, Junnian Wang, Yuzhen Liu, and Lei Jiang. An adaptive diversity introduction method for dynamic evolutionary multiobjective optimization. In *2014 IEEE Congress on Evolutionary Computation*, pp. 3160–3167, 2014. doi: 10.1109/CEC.2014.6900364.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

Thanh T. Nguyen, Shengxiang Yang, and Jürgen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

Juhyun Oh, Eunsu Kim, and Alice Oh. Flex-TravelPlanner: A benchmark for flexible planning with language agents. *arXiv preprint arXiv:2506.04649*, 2025. URL https://arxiv.org/abs/2506.04649.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013. doi: 10.1016/j.ejor.2012.08.015.

Junhao Qiu, Xin Chen, Liang Ge, Liyong Lin, Zhichao Lu, and Qingfu Zhang. Evolving interdependent operators with large language models for multi-objective combinatorial optimization, 2026. URL https://arxiv.org/abs/2601.17899.

Rindranirina Ramamonjison, Timothy T. Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. In *Proceedings of the NeurIPS 2022 Competition Track*, volume 220, pp. 189–203, 2023.

Shaaban Sahmoud and Haluk Rahmi Topcuoglu. Exploiting characterization of dynamism for enhancing dynamic multi-objective evolutionary algorithms. *Applied Soft Computing*, 85:105783, 2019. doi: 10.1016/j.asoc.2019.105783.

Andrej Schwanke, Lyubomir Ivanov, David Salinas, Frank Hutter, and Arber Zela. Multi-objective hierarchical optimization with large language models. *arXiv preprint arXiv:2601.13892*, 2026. URL https://arxiv.org/abs/2601.13892.

Yifan Shi, Jialong Shi, Jiayi Wang, Ye Fan, and Jianyong Sun. MIRROR: A multi-agent framework with iterative adaptive revision and hierarchical retrieval for optimization modeling in operations research. *arXiv preprint arXiv:2602.03318v1*, 2026. URL https://arxiv.org/abs/2602.03318v1.

Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2–9, 1989.

Raghav Thind, Youran Sun, Ling Liang, and Haizhao Yang. OptimAI: Optimization from natural language using LLM-powered AI agents. *arXiv preprint arXiv:2504.16918*, 2025.

Guangrui Xie. ORPilot: A production-oriented agentic LLM-for-OR tool for optimization modeling. *arXiv preprint arXiv:2605.02728*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27144–27152, 2025. doi: 10.1609/aaai.v39i25.34922. URL `https://ojs.aaai.org/index.php/AAAI/article/view/34922`.

Bowen Zhang, Pengcheng Luo, Genke Yang, Boon-Hee Soong, and Chau Yuen. OR-LLM-Agent: Automating modeling and solving of operations research optimization problems with reasoning LLM. *arXiv preprint arXiv:2503.10009*, 2025.

Chenyu Zhou, Xinyun Lu, Jiangyue Zhao, Jianghao Lin, Dongdong Ge, and Yinyu Ye. OR-Space: A full-lifecycle workspace benchmark for industrial optimization agents, 2026. URL `https://arxiv.org/abs/2605.28158`.

A Appendix

Appendix guide. The appendix first defines the mechanisms, metrics, and execution setup, then presents supplementary experiments. The Harness walkthrough follows the results. Prompt interfaces and the recorded benchmark requests appear last.

Topic	Go to	Evidence
Update execution	App. A.1–A.3	annotated algorithms, restart checks, and typed operators
Metrics and experimental setup	App. A.5–A.10	metrics, task definitions, interfaces, and numerical settings
Supplementary results	App. A.11–A.15	per-problem results, visual examples, controls, and sensitivity
Application walkthrough	App. A.16	a complete request-to-update walkthrough
Recorded prompts and requests	App. A.17–A.18	prompt interfaces, 15 initial requests, and 180 updates

Table 6: Appendix contents and the corresponding evidence locations.

A.1 MAPLE Initialization and Update Procedure

Algorithm 1 shows how MAPLE builds the initial program and handles each later request. LPD decides whether public data, the decision definition, or the evaluation function must change. TSS checks the two editable Workbench functions. LPD's `restart_skill` field is a provisional trace hint, not a population action. The semantic reuse assessment runs after the tentative Workbench passes compilation and the smoke test, then reads the accepted public history and a digest comparing the accepted and tentative states. A fixed gate checks structural support for the proposed Warm–Full choice. Its output replaces the provisional hint and becomes the effective restart action. The fixed optimizer then runs an exact solver, scalar evolutionary search, or Pareto evolutionary search. LSM stores events, the accepted state, and search archives with the public tables and Workbench. For a request about an earlier assignment, LPD identifies the requested relation. A fixed runtime procedure resolves the requested relation from stored events and accepted decisions, then enforces the binding during evaluation.

MAPLE w/o TSS builds its own representation, evaluation, search, and repair functions from the public `t00` task. It carries that Workbench, accepted state, and search archives through the sequence.

The fixed optimization loop still performs selection, maintains the archive, records metrics, and checks stopping. Table 10 specifies method inputs and carried state.

Algorithm notation. Let x_0 denote the initial request and u_t the t-th update. T is the number of updates. The repair limit B counts additional model attempts after a failed public check ($B = 3$ in the reported runs). The saved state is $S_t = (D_t, W_t, A_t, P_t, \hat{y}_t)$. E_t contains accepted update history. A_t is the returned archive, or a singleton for scalar tasks, and $\hat{y}_t$ is the online representative used for assignment lookup. Evolutionary search uses population P_t; the LP/MILP path does not use a population. A tilde marks tentative tables or functions that have not yet replaced the accepted state. The public-change summary Δ_t records differences between the accepted and tentative tables, decision declarations, and objectives. A smoke test runs a small example from the available input data. It checks data access, function calls, and output structure before numerical search.

Algorithm 1 MAPLE initialization and live update

Require: initial request x_0, initial public tables D_0, updates $u_{1:T}$, repair limit B
Ensure: accepted states and event histories (S_t, E_t), stopping at the first failed state

LLM: construct candidate Workbench $\widetilde{W}_0$ from (x_0, D_0). ▷ declare decisions and their evaluation
Compile and test $\widetilde{W}_0$ on a small public example. ▷ check signatures, types, and outputs
On public errors, **LLM:** repair and repeat these checks, at most B times.
if checks still fail **then**
    return failure. ▷ no initial state is committed
end if
Run the fixed solver; check execution and the public output contract.
if execution or output-contract checks fail **then**
    return failure. ▷ no post-search LLM repair
end if
Commit $S_0 = (D_0, W_0, A_0, P_0, \hat{y}_0)$ and start E_0. ▷ save the accepted result and initial event
for $t = 1, \ldots, T$ **do**
    LLM (LPD): identify edits and bindings required by u_t; emit trace hint h_t. ▷ h_t is provisional; no population is exposed
    Copy (D_{t-1}, W_{t-1}) into $(\widetilde{D}_t, \widetilde{W}_t)$. ▷ keep the committed state until acceptance
    Apply supplied table edits; **LLM:** translate language-only edits if needed.
    Pass h_t to the editor as non-operative trace context. ▷ restart logic is forbidden in the Workbench
    Edit only the requested Workbench functions (Algorithm 2).
    Compile and smoke-test; on public errors, allow at most B LLM repairs.
    if a check still fails **then**
        return failure. ▷ retain S_{t-1} and E_{t-1}
    end if
    Resolve queries using E_{t-1}, $\hat{y}_{t-1}$, and the revised decision domain.
    Log unresolved queries; retain only materialized bindings. ▷ unresolved queries remain inactive
    if the solver uses an evolutionary population **then**
        Compare old and candidate states to form Δ_t. ▷ changed fields, decision types, objectives
        LLM: assess reuse risk from u_t, public history, and Δ_t.
        $\alpha_t \leftarrow$ check the response with Algorithm 3. ▷ final Warm or Full action
        Set the effective action to α_t; record the original hint if it differs.
        Build the initial population for α_t. ▷ Warm: at most half repaired; Full: all fresh
    end if
    Wrap evolutionary candidate evaluation with the materialized assignment checks.
    Run the fixed solver and check the public output contract. ▷ fixed LP/MILP, GA, or NSGA-II
    if execution or output-contract checks fail **then**
        return failure. ▷ retain S_{t-1} and E_{t-1}
    end if
    Commit $S_t = (\widetilde{D}_t, \widetilde{W}_t, A_t, P_t, \hat{y}_t)$; extend E_{t-1} with the accepted event to obtain E_t.
end for

LLM calls are conditional on the requested data translation, function edit, or repair. Restart assessment uses the public update history and the accepted–candidate change digest after Workbench validation but before population construction. Held-out feasibility checks and reference scores are computed after execution.

Restart selection. The localizer provides a provisional hint h_t as context for the Workbench editor. After the edited functions compile and pass the smoke test, the semantic selector assesses reuse risk from a summary of changes between the accepted and candidate states. The fixed gate converts this assessment to $\alpha_t \in \{\text{WARM}, \text{FULL}\}$, which replaces h_t and determines the initial population for search.

Algorithm 2 Updating the two TSS Workbench functions

Require: LPD edit labels, candidate public tables $\widetilde{D}_t$, accepted Workbench W_{t-1}
Ensure: checked candidate Workbench $\widetilde{W}_t$, or a public error for Algorithm 1

$\widetilde{W}_t \leftarrow \text{copy}(W_{t-1})$ ▷ preserve functions outside the requested edit
if LPD requests a decision-definition edit **then**
    LLM: edit `build_problem()`. ▷ data, domains, solver, or linear formulation
end if
if LPD requests a candidate-evaluation edit **then**
    LLM: edit `evaluate()`. ▷ evolutionary objectives, constraints, and decoding
end if
Bind declared types to their fixed operators. ▷ initialization, crossover, mutation, domain repair
Compile; check solver choice, objective order, and required outputs.
Execute a small example using $\widetilde{D}_t$. ▷ exercise data access, decoding, and evaluation
if any public check fails **then**
    return the public error to Algorithm 1. ▷ the outer loop owns the bounded repair budget
end if
return $\widetilde{W}_t$. ▷ LP/MILP, GA, and NSGA-II routines remain fixed

Executing historical assignment bindings. After Warm or Full constructs the initial candidates, a fixed wrapper checks every candidate evaluation against all resolved assignment locks. A mismatch adds a violation and a scalar penalty while leaving the objective vector unchanged. Crossover and mutation therefore remain type-based, and feasibility-first selection handles binding violations. The binding assay additionally checks the returned archive against the requested locks and assigns zero to missing or incorrect bindings.

A.2 Checks for Restart Recommendations

The model proposes whether to reuse earlier candidates. The fixed checks determine whether the response meets the conditions for a fresh start.

Algorithm 3 Fixed gate for semantic restart selection

Require: public-change summary Δ_t, model assessment (r, M, E, v)
Ensure: restart action Warm or Full

$V \leftarrow M \cap \text{SupportedMechanisms}(\Delta_t)$ ▷ check structural support
$C \leftarrow E \cap \text{ChangedFields}(\Delta_t)$ ▷ verify the cited changes
return Full if $r =$ `high` and $v =$ `true` and $V \neq \varnothing$ and $C \neq \varnothing$;
    otherwise return Warm.

Here r is the assessed reuse risk, M contains proposed causes, E contains cited public changes, and v is the model's Full vote. The gate checks category support and changed paths separately. Table 7 gives the structural tests.

Parsing the model response. The parser lowercases the risk string and trims its whitespace. It removes unrecognized mechanisms and duplicate list entries. It also strips whitespace and outer slashes from cited field paths. A cited path is retained only if it belongs to the public change summary's changed-field set. The five allowed identifiers are `decision_support_replacement`, `objective_preference_reversal`, `constraint_regime_change`, `resource_role_reversal`, and `distant_basin_risk`. Their checks are listed below.

Recognized constraint fields are `active`, `available`, `availability`, `capacity`, `cpu`, `mem`, `gpu`, `gpu_required`, `max_shifts`, `required`, `eligibility`, `eligible`, and `shift_end`. Resource tables are `machines`, `vehicles`, `nurses`, `staff`, and `resources`; their recognized fields are `capacity`, `cpu`, `mem`, `gpu`, `emission_rate`, `energy_idle`, `energy_per_cpu`, and `max_shifts`. Preference paths con-

Mechanism	Required structural change in the public summary
Decision-support replacement	Changed typed segments; or active-ID Jaccard overlap at most 0.5; or row-ID overlap at most 0.5 in a table with at least four old and four new rows.
Objective-preference reversal	Both a preference path and a policy path change; a changed evaluation function can supply the policy path.
Constraint-regime change	Changed typed segments, active-ID overlap at most 0.5, or at least two changed paths ending in a recognized constraint field.
Resource-role reversal	At least two changed resource-table paths ending in a recognized capacity, compatibility, emissions, or energy field.
Distant-basin risk	At least one of the preceding four structural tests passes.

Table 7: Structural support tests used by the fixed restart gate. The model must also report high risk, cast a Full vote, and cite at least one changed path.

tain `/preferences/` or `/preference`. Policy paths contain `/policy/` or equal `workbench/fitness`. These tests use the listed table and field mappings.

The path and mechanism tests are separate. The `reason` string is logged and does not enter the gate. The prompt requests a JSON boolean for `full_vote`. The implementation applies Python boolean conversion. Missing fields have defaults of unknown risk, empty lists, and a false vote. Ordinary parsing or provider errors return Warm. A detected quota-limit error is passed back to the caller.

A.3 TSS Workbench and Available Choices

The TSS Workbench is a reusable optimization program. A small catalog describes the decision types and solvers available to the Workbench-building model, together with the fixed checks and restart actions available to the runtime. The model edits only the decision-definition function and the evaluation function. The first reads public tables, declares typed decisions, chooses the solver, and names the outputs. The second checks one candidate against the public constraints and objectives. Table 8 summarizes this boundary.

How linear problems are sent to the solver. The Workbench specifies variables, bounds, the objective, and linear constraints for the fixed solver. Each variable has a name and a continuous or integer domain. The objective declaration includes coefficients and a direction. Each constraint supplies a coefficient row, a relation, and a right-hand side. The fixed backend builds the coefficient arrays and uses HiGHS through SciPy's linear or mixed-integer solver. A declared extraction rule maps returned values to selected entities, assignments, or schedules. The runtime validates that output and retains the solver status and objective. The formulation returned by `build_problem()` under `linear_program_spec` supplies the model to the fixed linear solver. Changes to linear variables, objectives, or constraints edit this declaration or the public tables it reads.

Adding a decision type or solver. Each catalog entry specifies its inputs, outputs, use conditions, and possible failures. An entry can call an existing solver or restart policy. Adding a decision type requires implementing its initialization, variation, and repair operators and registering them in the catalog.

A genome composes one or more typed segments; composition is not an additional primitive type. TSS rejects a decision space that cannot be expressed by its declared decision types. Solver selection preserves the public objective names and cannot turn a Pareto task into a scalar one. Warm migrates historical candidates into the current decision domains and re-evaluates their feasibility. It deduplicates the repaired seeds and fills the remaining population slots with fresh candidates.

How old candidates are repaired before search. Warm first adapts earlier candidates to the updated decisions and checks their feasibility. These candidates come from the previous population and archive. For multiple objectives, the seed ordering includes objective extremes, knee candidates, and nondominated selection with crowding. Each candidate is first migrated to the current domains, then evaluated. The local search tests type-specific moves and accepts the first improvement in the pair $(\mathbf{1}[\text{infeasible}], f_{\text{scalar}})$. It stops after a sweep with no improvement or 80 evaluated proposals.

Public-data change: P015-t02

```
tables/machines[id=M04]
- energy_per_cpu: 0.23
+ energy_per_cpu: 0.285
```

The remaining public fields and both editable functions are unchanged.

Evaluation-function change: P010-t02

```
--- Before update
+++ After update
@@ congestion anchor @@
+    congestion_anchor_set = set()
+    for oid, o in data["order_by_id"].items():
+        if abs(o["x"] - 48) < 1e-9 and abs(o["y"] - 11) < 1e-9:
+            congestion_anchor_set.add(oid)
+            break   # at most one anchor
@@ route traversal @@
+        prev_oid = None   # no order before first stop
+            if prev_oid in congestion_anchor_set or oid in congestion_anchor_set:
+                leg_dist *= 1.25
+            prev_oid = oid
@@ return-to-depot leg @@
+        if prev_oid in congestion_anchor_set:
+            leg_dist *= 1.25
```

Excerpt from the saved patch; unrelated lines are omitted.

Figure 6: **Two localized Workbench edits.** The left patch changes public data. The right patch changes route evaluation using a coordinate-based congestion anchor and a distance multiplier. Appendix A.6 examines the historical binding and the travel-time calculation required by the public request.

Choice	Available options	When code must be added
Decision types	binary, categorical, integer, real, permutation, assignment	a new decision type
Solvers	LP/MILP, scalar EA, Pareto EA	a new solver
Population start	Warm (repaired history plus fresh candidates) or Full (fresh candidates)	a new restart action
Checks	compilation, output fields, feasibility, and archive checks	a new safety rule

Table 8: Choices exposed by the TSS runtime. The Workbench-building model selects decision types and a solver route. Fixed code supplies checks and operators, while the separate semantic assessment proposes Warm or Full after an update.

The standard fitness helper computes

$$f_{\text{scalar}}(x) = f_{\text{base},W}(x) + \lambda_W V_W(x),$$

where the generated evaluator supplies the base scalar and violation values. The helper reduces violations to V_W and uses $\lambda_W = 10^6$ by default. Repair uses this scalar without additional runtime normalization, while subsequent NSGA-II selection uses the separate objective vector. After deduplication, seed feasibility is checked again for reporting. The returned seeds enter feasibility-first search, and the remaining slots are sampled afresh.

With population 200 and at most 100 historical candidates, Warm uses up to 8,200 additional evaluation calls before the shared evolutionary loop. The loop uses $N(g+1)$ calls for population size N and g completed generations. Reported generation counts measure this loop, while stage wall time also includes initialization, migration, repair, and validation.

A.4 Limitations and Ethics

MAPLE relies on a fixed catalog of decision types, solver routes, and restart actions. Tasks requiring a new decision type or numerical operator need a runtime extension. Typed operators preserve representation membership, but feasibility still depends on the generated evaluation function.

Unresolved historical binding queries are logged and skipped, so only successfully resolved bindings constrain search. LP/MILP revisions rely on LPD selecting the formulation-edit path. The semantic restart gate depends on the model's interpretation and fixed field mappings for some structural checks.

The evolutionary routes optimize within finite budgets. Repairing retained candidates adds work before search, so the choice between reuse and fresh initialization also depends on the available latency.

The current implementation runs generated Python functions in the application process. Operational use requires stronger isolation of files, imports, credentials, and network access. MAPLE is intended for batch or human-supervised re-optimization. High-impact decisions should support inspection, human approval, and rollback.

A.5 Metric Definitions

NLDO evaluates every state with a fixed hidden evaluator. For scalar profiles, let $f_t(x)$ be the evaluation objective and let $f_t^\star$ be the exact or best-known reference after sign conversion. The normalized scalar quality is

$$Q_t^{\text{scalar}} = \mathbf{1}[\text{feasible}] \max\left(0, 1 - \frac{\max(0, f_t(x) - f_t^\star)}{\max(|f_t(x)|, |f_t^\star|, 1)}\right).$$

An infeasible output receives zero. An output that matches the reference receives one. Scalar tables use bounded 0–1 quality. Pareto tables use the raw ratio below. Dynamic tables separate *solve rate* from *online quality* using the quantities in Section 4. The recorded indicator a_t includes the original output-protocol and validity decisions, and r_t marks their consecutive accepted prefix. The mathematical check v_t requires at least one feasible plan in the saved output. Solve rate averages $r_t v_t$, and online quality averages $r_t q_t$. An output may therefore be mathematically feasible but ineligible because its original protocol failed or an earlier state was rejected. Unattempted states have $v_t = q_t = 0$ and remain in the denominator. The archive scorer computes mathematical validity, while the trajectory aggregation applies recorded eligibility. For MAPLE, we average quality across ten search seeds at each state, then across states. The reported numerical variation is computed from the seed means. The DS denominator is 9 scalar episodes $\times$ 13 sequential states and the DM denominator is 6 Pareto episodes $\times$ 13 sequential states. Analyses restricted to updates have denominators 108 and 72, respectively, and are explicitly labelled t01–t12. Online quality is the mean quality over those same states, with infeasible and unsolved states contributing zero.

For multi-objective profiles, the evaluator compares a submitted archive A_t with a withheld reference archive R_t after removing infeasible candidates. For every multi-objective episode and time step, R_t is built from 10 hidden reference runs. Each run uses population 500 and 500 generations. We pool all ten final populations, recheck feasibility, and keep the nondominated set without truncating the merged front. Each time step therefore starts from 5,000 reference candidates before filtering and deduplication. The 78 Pareto-state references cover initialization and all updates under the public objective definitions. This construction is independent of any submitted method. It uses the same public state as the submitted methods but a larger search budget. The reference search has no early stopping and starts independently at each state. We retain each seed's final population, random seed, completed generation count, and checksums. All objectives are minimized. Each time step uses fixed ideal and reference points derived from its held-out archive. The submitted and reference archives use the same points. For objective i, let $\ell_{t,i}$ and $h_{t,i}$ be its minimum and maximum over R_t. Let $s_{t,i}$ be the bound stored in the reference archive's `hypervolume_reference_point` metadata. The reference builder uses a margin of 0.15 for that stored bound. Evaluation computes a wider bound $c_{t,i}$ and uses the effective bound $b_{t,i}$:

$$c_{t,i} = \text{round}_6\left[h_{t,i} + 0.5 \max(h_{t,i} - \ell_{t,i}, |h_{t,i}|, 1) + 10^{-9}\right],$$
$$b_{t,i} = \text{round}_6[\max(s_{t,i}, c_{t,i})].$$

Here round_6 rounds to six decimal places. If stored metadata is absent, $b_{t,i} = c_{t,i}$. Both archives use coordinates

$$\hat{f}_{t,i} = \text{clip}_{[0,1.5]}\left(\frac{f_{t,i} - \ell_{t,i}}{\max(b_{t,i} - \ell_{t,i}, 10^{-9})}\right).$$

Here $\text{clip}_{[0,1.5]}(z) = \min(1.5, \max(0, z))$. The bounds are shared by every method at that stage and do not change as its search proceeds. An objective value better than the ideal point is clipped to that point. A value worse than the reference bound contributes no volume inside the unit HV box. IGD uses these same clipped coordinates, including the upper cap of 1.5. The benchmark has two or three objectives, for which the runtime computes HV exactly in this normalized box. We report hypervolume ratio

$$\text{HVRatio}_t = \frac{\text{HV}(A_t)}{\text{HV}(R_t)}$$

without clipping. Because R_t is a finite best-known set, a feasible archive may slightly exceed a ratio of one. The evaluator removes infeasible candidates before computing HV and IGD. An archive with no feasible candidate receives zero HV ratio. IGD is reported only when an eligible archive contains at least one feasible candidate. We also report inverted generational distance

$$\mathrm{IGD}_t = \frac{1}{|R_t|} \sum_{y \in R_t} \min_{x \in A_t} \|\hat{\mathbf{f}}_t(x) - \hat{\mathbf{f}}_t(y)\|_2.$$

Ideal gap uses a separate anchor $z_{t,i} = \ell_{t,i} - 0.1 \max(|\ell_{t,i}|, 1)$ and reports the closest feasible candidate's scaled Euclidean distance:

$$\mathrm{IdealGap}_t = \min_{x \in A_t} \sqrt{\sum_i \left[\max\left(0, \frac{f_{t,i}(x) - z_{t,i}}{\max(b_{t,i} - z_{t,i}, 10^{-9})}\right)\right]^2}.$$

Smaller is better. Unlike HV and IGD coordinates, this distance has no upper clipping. The implementation rounds it to six decimal places and returns infinity for an empty archive, which is omitted from finite-distance summaries. Tables report raw HV ratio, IGD, and ideal gap. The primary saved-plan evaluator rechecks all submitted candidates without imposing a new archive-size cap. Dominated or duplicate points add no HV. Any archive cap used while producing a method's saved output belongs to that method's search, not to the pooled reference construction.

A.6 Benchmark and Evaluation Details

NLDO has five profiles with three episodes each, spanning established selection, scheduling, rostering, routing, and cloud-placement settings (Beasley, 1990; Brandimarte, 1993; Ceschia et al., 2018; Pillac et al., 2013; Erdoğan & Miller-Hooks, 2012; Bektaş & Laporte, 2011; Beloglazov et al., 2012). Compact scalar tasks use exact references, larger scalar tasks use best-known solutions, and Pareto tasks use held-out archives.

Effects of the requested updates. We label updates by changes to feasibility, candidate rankings, objective scale, or reported diagnostics. A changed table does not always imply changed optimal decisions. Multiplying one objective by a candidate-independent positive constant preserves Pareto dominance, while changing a coefficient for only one machine or vehicle can reorder candidates. A deadline used only in a reported diagnostic has a different role from a hard return deadline. Historical reference is a separate annotation.

View	Setting	Objectives	Episodes	Time	Domains
SS	Static	Single-objective	P001–P009	t00	selection/allocation, precedence scheduling, coverage rostering
SM	Static	Multi-objective	P010–P015	t00	ordered service routing, cloud-resource placement
DS	Dynamic	Single-objective	P001–P009	t00–t12	selection/allocation, precedence scheduling, coverage rostering
DM	Dynamic	Multi-objective	P010–P015	t00–t12	ordered service routing, cloud-resource placement

Table 9: The four NLDO views and their problem, time, and domain coverage. Dynamic views include initialization. Update-only analyses exclude t00 and are labelled separately.

Each profile contributes 36 updates. Thirty of the 180 updates require information from earlier requests. Across all updates, 103 first paragraphs have distinct wording. Rostering, routing, and cloud placement contribute six disruptive t11–t12 states each. All table changes retain the initial columns and listed decision entities.

Agent inputs and evaluation annotations. The agent receives the problem description and input tables. Separate annotations identify the intended entities and reference chains for evaluation. Table 10 defines method inputs and retained information.

How saved routes and assignments are scored. We recompute the objectives of saved plans using the formulas stated below.

Routing travel time uses distance divided by speed. Reducing speed by 25% multiplies the affected travel time by $4/3$, leaving physical distance unchanged. Emissions sum route distance times vehicle

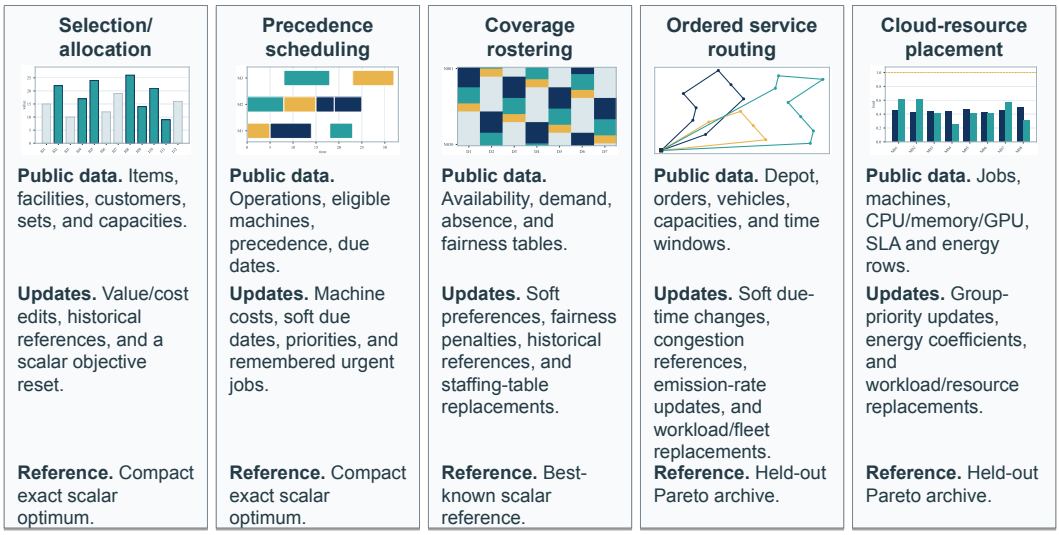


Figure 7: **Five NLDO problem families.** Each card pairs a representative MAPLE solution with the public data, update types, and reference used for that family. The tasks span compact exact problems, larger scalar search, and Pareto routing and placement.

emission rate and the carbon multiplier. The public definition applies congestion only on arrival at the named order, with its identity retained across later edits to that order. Every nonempty route leaves the current depot at time zero and returns by the vehicle's shift end. Waiting until an order's ready time and its service duration both count toward elapsed time. Lateness measures service-start time beyond the soft due time. A requested reduction of a soft due target applies to that target directly. It can precede the ready time, in which case the earliest service already incurs lateness. For example, reducing a soft due target of 87 by 50 gives 37. With a ready time of 40 and a revised soft due time of 37, the earliest service incurs three units of lateness.

Cloud energy sums one idle charge per used machine and the assigned CPU demand times that machine's per-CPU energy rate, scaled by energy price and carbon intensity. The hidden evaluator measures imbalance as

$$B = 100 \sum_{m \in \mathcal{M}_{\text{avail}}} |u_m - \bar{u}|, \qquad u_m = \frac{\sum_{j:a(j)=m} c_j}{C_m}, \qquad \bar{u} = \frac{\sum_{m \in \mathcal{M}_{\text{avail}}} u_m}{|\mathcal{M}_{\text{avail}}|}.$$

Here c_j is an active job's CPU demand and C_m is machine capacity. Available but unused machines contribute zero utilization to the mean.

Evaluation protocol. The agent comparison, complete TSS ablation, full-sequence restart comparison, and separate Kimi run evaluate saved plans under the objective definitions above, using common state-specific references. Programs and plans come from the recorded agent runs, with objective values computed independently from the submitted routes and assignments. Single-update restart, search-operator, independent-model, binding-replay, and reference-sensitivity controls retain their recorded evaluation settings, specified with the corresponding results. Comparisons within each control use the same evaluator and reference set.

A.7 Static Solving Setup

The static supplement uses NLP4LP-hard and BWOR20 to measure ordinary one-shot optimization from language. These tasks contain no updates, memory, restart choice, or predecessor population. Each row provides the problem text and public parameters. Each method produces executable Python solver code and follows its published process when available.

Evaluation of initial solutions. We report whether each program runs and whether its returned objective matches the reference. Table 2 reports execution rate and Objective, the fraction of all cases whose returned objective agrees with the reference under the recorded numeric tolerance.

Objective agreement accepts alternative optimal decision vectors. The comparison accepts $|\hat{f} - f^\star| \leq \max(\epsilon_a, \epsilon_r|f^\star|)$, including equality at the threshold. NLP4LP-hard uses $\epsilon_a = \epsilon_r = 10^{-6}$. The BWOR20 manifest overrides these defaults with $\epsilon_a = 0.1$ and $\epsilon_r = 0$. An output without a predicted objective fails the objective check. Status-only acceptance is recorded separately. Both marginal rates are available for all seven methods, with denominators of 59 NLP4LP-hard cases and 20 BWOR20 cases.

Additional diagnostic checks. We separately report compilation, execution, objective agreement, and the original output checks. For case i, let c_i and e_i denote compilation and execution, o_i reference-objective agreement, and s_i correct reference status on a status-only case. The joint Match diagnostic in Tables 39–40 is

$$\frac{1}{N}\sum_{i=1}^{N} c_i e_i \mathbf{1}[o_i \text{ or } (\text{status-only case } i \text{ and } s_i)] .$$

Its denominator includes every benchmark case, including execution failures. The tables also retain compilation, execution, objective agreement, and the original workflow-specific output and named-variable checks. These diagnostics can overlap. Dashes denote unreported supplementary checks. Joint Match is unreported for ReAct and OptimAI.

Selection of the static benchmark cases. NLP4LP-hard and BWOR20 use fixed subsets of the released datasets. NLP4LP-hard comes from the NLP4LP dataset released with OptiMUS (AhmadiTeshnizi et al., 2024). Our saved snapshot contains 361 instances, including 65 labelled hard. We retain the 59 hard instances with a nonempty reference solution. The other six have no value in the solution field. The reference objectives are read from each retained instance's supplied `solution` record. BWOR20 is our fixed 20-case sample from the 82-row BWOR collection distributed with OR-LLM-Agent (Zhang et al., 2025). The manifest samples without replacement using Python's random generator with seed 20260621, then sorts the selected case identifiers. Its references use the supplied `answer` field.

A.8 Baseline Interfaces and Execution

Code and plans retained by Persistent ReAct. Persistent ReAct retains its preceding solver code and accepted answer, then edits the program for the next request. It is our stateful control based on the ReAct pattern. At initialization, the model writes a complete Python solver program using public parameters and solver libraries. At an update it receives the initial public tables, cumulative dialogue, and its carried workspace, then emits the revised program, which applies the requested changes before solving.

The runtime parses, compiles, and executes the program, then reads its printed solution. Compilation, runtime, or solution-format failures return the actual error and available output to the model for at most three code-output repairs. The reasoning and tool-plan fields are auxiliary outputs. Missing auxiliary fields are checked separately and did not trigger code repair in the saved runs.

The workspace retains the complete preceding solver code and a compact final answer. A separate history field contains the preceding plan and cumulative updates. Public execution success with an extracted solution advances this workspace. Nothing is carried at t00. A program may produce multiple candidates and a Pareto set within a stage; the cross-stage workspace retains code and answers, with no separate population or Pareto archive.

A.9 Numerical Settings and Released Materials

Table 11 lists the settings used in the experiments. Main model calls share a provider and temperature, with output limits determined by the requested operation. The table separates repair limits from evolutionary-search settings. Stopping rules and budgets were set on pilot material.

Method	Implementation	Public inputs	Code retained for the next request	Accepted plan	Cross-stage population/ archive	Repair feedback
MAPLE	Typed Workbench runtime	Full history	Two functions	Own representative	Own population and archive	Compilation and small-example errors
w/o TSS	Untyped Workbench runtime	Full history	Full artifact	Own representative	Own population and archive	Compilation and small-example errors
Persistent ReAct	Stateful executed-code control	Full history	Complete solver	Own compact answer and plan	None separate from answer	Compile, runtime, output
ReAct	Public-code adapter	Full history	In own dialogue, if present	Own, on continuation	Own, if returned on continuation	Compile, runtime, output
ORLM / OptiMUS / OR-LLM-Agent	Workflows based on the named methods	Full history	In own dialogue, if present	Own, on continuation	Own, if returned on continuation	Compile, runtime, output
OptimAI	Paper-based reproduction	Full history	In own dialogue, if present	Own, on continuation	Own, if returned on continuation	Compile, runtime, output

Table 10: **Executed NLDO interfaces.** Cumulative inputs include the initial request, public tables, required outputs, and requests through the current stage. Each external workflow uses its own history, with at most three code-output repairs.

Setting	Value	Applies to
Language model	DeepSeek-V4-Pro (Preview), API ID `deepseek-v4-pro`, temperature 0.0	all main model calls
LLM output limits	1200–18000 tokens	depend on the requested task
Workbench edit repair	3 attempts	MAPLE Workbench checks
Workbench check	compilation and hard-coded-ID check	public table identifiers only
External-code repair	3 attempts	external methods
Search budget	population 200, at most 200 generations, seeds 0–9	main numerical runs
Search archives	at most 500 feasible, unique, nondominated points	main numerical runs
restart selector	semantic choice from stored events and the public change summary	no trial search or hidden score
Restart actions	Warm: up to 100 repaired, then fresh. Full: 200 fresh	same solver and search budget
Scalar stopping	start at g=40, gain $\epsilon = 5\times10^{-4}$, patience 25	feasible scalar objective
Pareto stopping	g=40, min. archive 8, $\epsilon = 5\times10^{-4}$, box 0.01	fixed internal objective box
Metric record	every generation, at most 200 saved points	recording only
Held-out references	fixed 500×500, 10-seed set per time step	final metrics only
Model repeats	3 Workbench sequences $\times$ seeds 0–2	P010 routing and P015 cloud, fixed restart choices

Table 11: Settings for the reported experiments.

Stopping rule. Main evolutionary runs stop after at most 200 generations. From generation 40 onward, a run may stop after 25 generations without sufficient improvement. A scalar run requires a relative improvement above 5×10^{-4} in its best feasible objective. A Pareto run becomes eligible only with at least eight feasible archive members. The run's objective box is then fixed with a 5% upper margin. The 25-generation count restarts after an internal HV gain above 5×10^{-4}. It also restarts after the front occupies a new normalized cell of width 0.01 or improves the normalized ideal point by more than 5×10^{-4}. Stopping uses these internal Workbench signals. Held-out HV and IGD are computed from saved candidates for final scoring and generation-wise plots.

A.10 Reproducibility Materials

The supplementary materials contain the public requests and tables, prompts, generated Workbench examples, numerical settings, and saved result records. The evaluation package provides versioned scoring code and state-specific references for scoring saved plans. Aggregation combines these scores with the recorded acceptance prefixes.

The evaluation package contains 195 states, scalar references, and 78 Pareto references from ten 500×500 searches per state, with reference manifests and file checksums. The primary aggregate uses 4,368 Pareto state–seed records, original acceptance indicators, and the retained static execution/objective summaries. Supplementary control records include P007–P015 matched restarts, 180 update transitions, 30 historical references, state-binding cases, independent P010/P015 model runs, and reference/archive checks.

Episode-bootstrap intervals use 20,000 resamples (NumPy generator seed 20260905).

The DeepSeek-V4-Pro (Preview) checkpoint used in these experiments is no longer available from the provider.

A.11 Aggregate and Per-Problem Results

Routing and cloud-placement results. Table 12 separates results for the two Pareto domains.

Method	Routing HV	Cloud HV	Overall HV	IGD ↓
MAPLE	0.768	0.982	0.875	0.082
Always Warm	0.749	0.982	0.866	0.093
Always Full	0.665	0.984	0.825	0.112
w/o TSS	0.394	0.936	0.665	0.143

Table 12: **Public-formula evaluation of saved Pareto plans.** All rows cover t00–t12 and ten numerical seeds. Fixed policies follow their own histories. HV includes zero-valued invalid states. IGD is conditional on valid archives, with 780 states for each full-system policy and 630 for w/o TSS.

MAPLE's overall HV is 0.875, compared with 0.866 for own-history Warm and 0.825 for own-history Full. The domain breakdown locates the gain: adaptive reuse helps routing, while Full is slightly stronger in cloud placement. The untyped Workbench can find strong cloud plans but loses validity on routing.

The following summaries cover all 15 episodes with population 200, at most 200 generations per state, and 10 seeds.

Effect of auxiliary output fields. We recompute the eligible prefix while ignoring auxiliary fields and keeping the saved plans and feasibility checks unchanged. Table 17 includes every saved attempt at t01–t12, including attempts after an earlier output-check failure. Persistent ReAct's initial P004 program runs, but its schedule is infeasible and its output lacks a required reasoning field. Its accepted prefix therefore has length zero. Table 18 lists three feasible rosters rejected because required auxiliary fields were missing. Those omissions did not trigger code repair.

Episode	Profile	HV mean ± SD	IGD mean ± SD
P010	Ordered service	0.754±0.076	0.154±0.051
P011	Ordered service	0.759±0.070	0.150±0.040
P012	Ordered service	0.790±0.052	0.157±0.040
P013	Cloud resources	0.986±0.004	0.011±0.004
P014	Cloud resources	0.972±0.003	0.013±0.001
P015	Cloud resources	0.989±0.003	0.010±0.001

Table 13: **Public-formula Pareto quality by episode.** Every row uses 10 seeds and 13 states per seed against the same freshly searched reference union. HV and IGD first average over time steps and then report variation across seeds.

DS			DM			
Update type	**States**	**Online quality**	**Update type**	**States**	**Online quality**	**IGD ↓**
All updates	108	0.947	All updates	72	0.879	0.079
Feasibility	3	0.721	Feasibility	15	0.722	0.163
Objective ordering	63	0.942	Objective ordering	48	0.836	0.104
Positive rescaling	0	–	Positive rescaling	12	0.940	0.044
Diagnostic only	6	1.000	Diagnostic only	12	0.988	0.010
Unchanged	39	0.948	Unchanged	0	–	–
Historical reference	21	0.996	Historical reference	9	0.894	0.070

Table 14: **MAPLE results by audited update semantics**, t01–t12. Ten-seed state means use the original scalar evaluation for DS and public formulas for DM. Labels may overlap; historical reference is a separate dimension. Diagnostic-only changes preserve objectives and hard constraints. Solve rate is 100% in every nonempty group; empty groups have unavailable scores.

NLDO profile	Method	Overall mean	Dynamic mean	Gap/HV	IGD ↓
Selection/allocation	MAPLE (ours)	0.957	0.953	19.231	–
Precedence scheduling	MAPLE (ours)	0.965	0.962	11.038	–
Coverage rostering	MAPLE (ours)	0.933	0.927	298.920	–
Ordered service routing	MAPLE (ours)	0.752	0.757	0.757	0.165
Cloud-resource placement	MAPLE (ours)	0.805	0.801	0.801	0.129

Table 15: **Recorded-evaluator results by profile.** Overall includes t00. Dynamic covers t01–t12. Scalar rows report objective gap. Pareto rows report HV and IGD. Public-formula results are in Table 13.

For every external method, the mathematical prefix $\tilde{r}_t = \prod_{j=0}^{t} v_j$ accepts feasible plans regardless of auxiliary fields, retaining the same plans, public formulas, references, and zero missing suffixes. Persistent ReAct continued running after some auxiliary fields were omitted. Its scalar solve rate rises from 51.3% to 85.5%, and its Pareto solve rate from 11.5% to 42.3%. Table 19 provides the paired rates for all external methods.

A.12 Case Study and Visual Result Atlas

Figures 8 and 9 show how accepted plans, Pareto fronts, and search traces change across successive routing and cloud-placement requests.

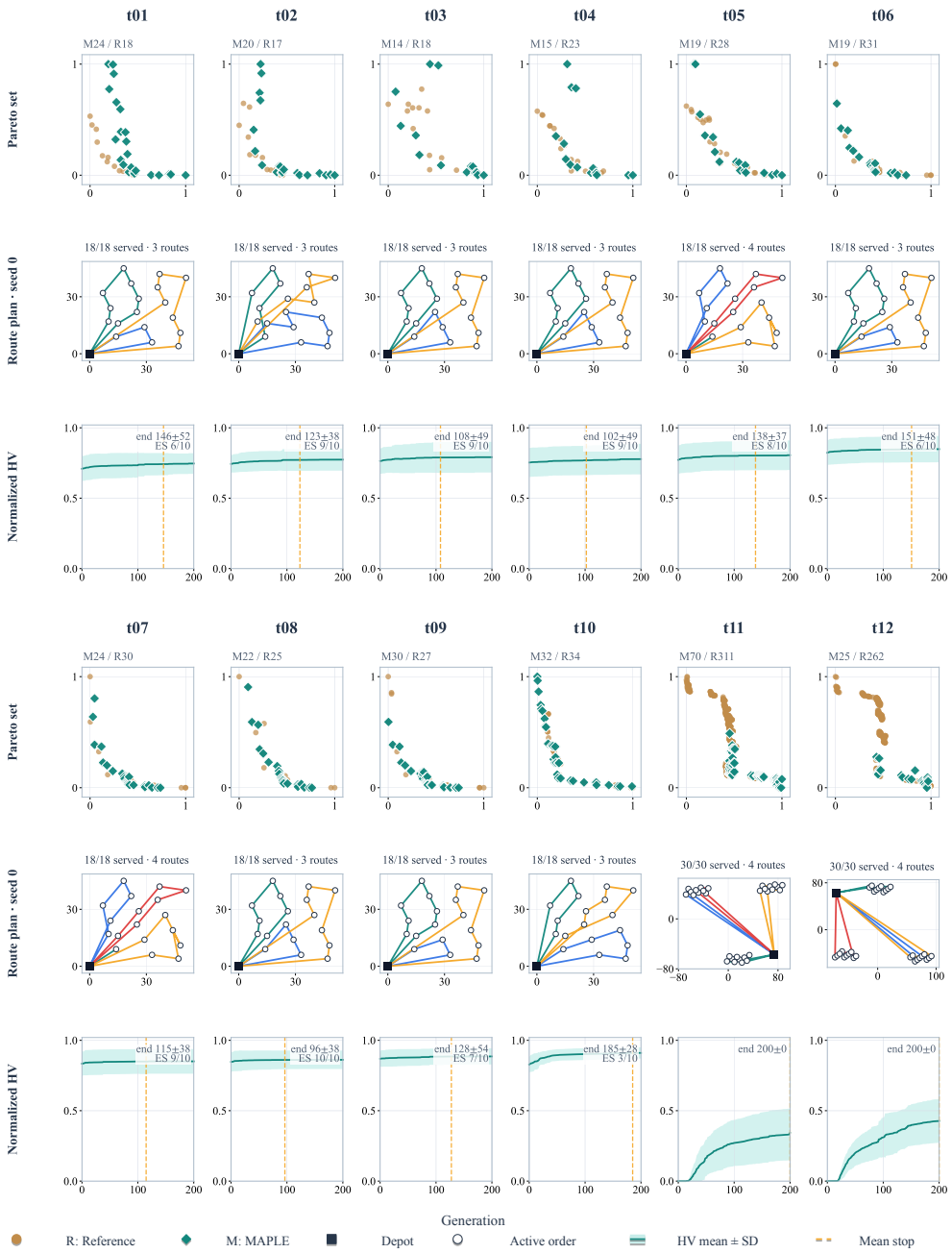


Figure 8: **Ordered-service routing across t01–t12 (NLDO-P010).** The upper/lower blocks show t01–t06/t07–t12. Teal diamonds show MAPLE's ten-seed pooled Pareto set; amber circles show the recorded reference. Route panels show the seed-0 plan, zoomed to active orders; inactive orders are omitted. Front coordinates normalize distance and lateness within each stage. HV uses all three objectives, including emissions.

The third row follows normalized HV during search. Generation 0 is the population just after restart. Curves remain constant after termination. The dashed line marks the mean stopping generation.

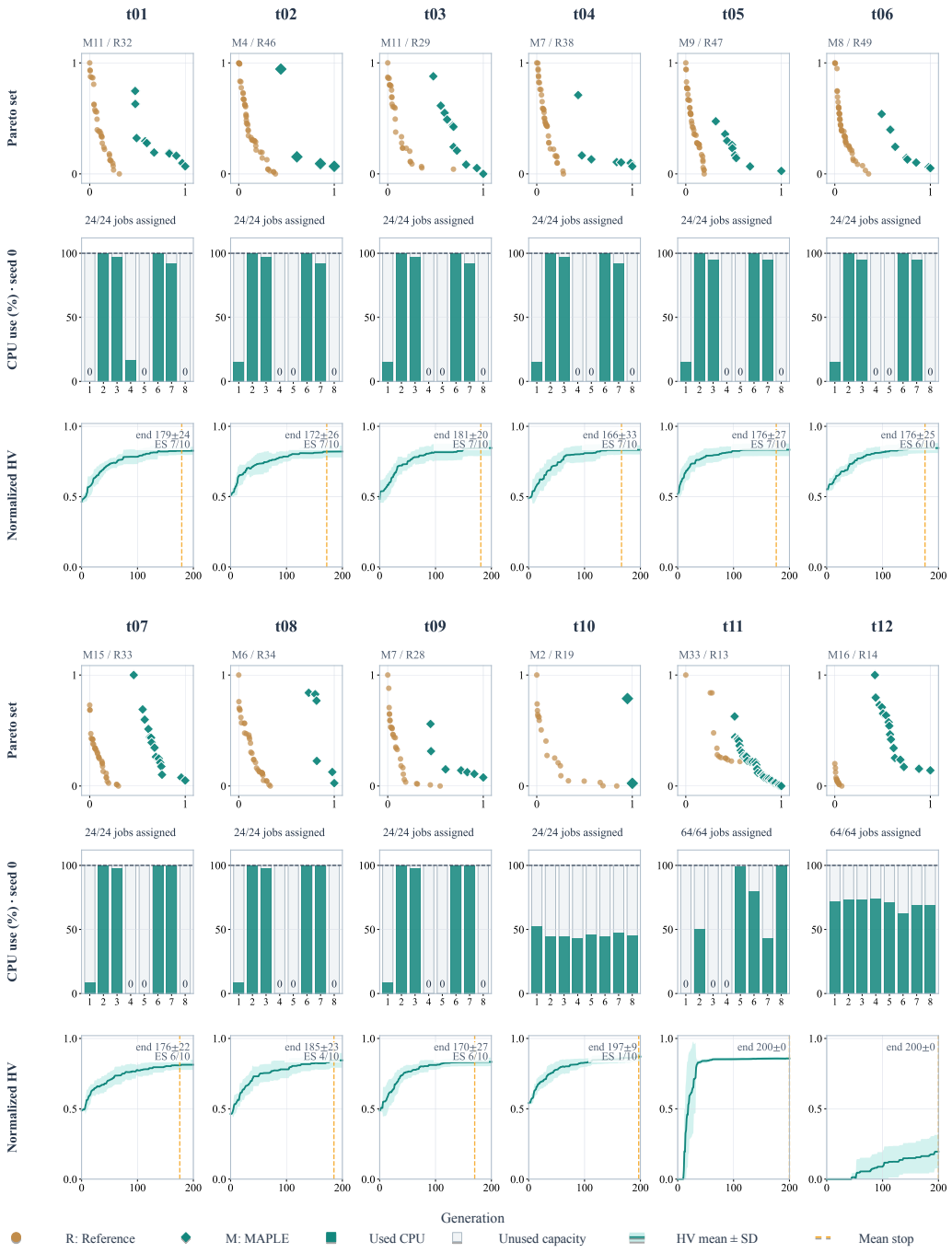


Figure 9: **Cloud-resource placement across t01–t12 (NLDO-P015).** Blocks show t01–t06 and t07–t12. Diamonds show MAPLE's ten-seed pooled energy–load-imbalance fronts; circles show the reference, normalized within each stage. Seed-0 plans show CPU utilization on machines 1–8: teal marks usage, pale bars mark unused capacity, and 0 marks idle machines. Ten-seed HV curves show the initial quality loss and recovery at t11/t12.

Accepted plans with lower solution quality. We inspect accepted plans whose normalized quality decreases after a revision. Accepted outputs pass the runtime's public checks, but their evaluated quality can still be low. At P002-t10, the scalar objective changes from monetary cost to service time. The retained assignment remains feasible and has lower normalized quality under the revised objective. At cloud update t12, the machine identifiers remain the same, but workloads and resource properties change. A feasible placement can have lower quality under the revised problem.

Episode	Profile	Quality or HV	Obj./ideal gap ↓	HV	IGD ↓
P001	Selection/allocation	1.000	0.000	–	–
P002	Selection/allocation	0.871	57.692	–	–
P003	Selection/allocation	1.000	0.000	–	–
P004	Precedence scheduling	0.894	33.115	–	–
P005	Precedence scheduling	1.000	0.000	–	–
P006	Precedence scheduling	1.000	0.000	–	–
P007	Coverage rostering	0.930	452.043	–	–
P008	Coverage rostering	1.000	-7.873	–	–
P009	Coverage rostering	0.868	452.592	–	–
P010	NLDO-DM, ordered service	0.747	0.415	0.747	0.165
P011	NLDO-DM, ordered service	0.733	0.417	0.733	0.187
P012	NLDO-DM, ordered service	0.777	0.407	0.777	0.157
P013	NLDO-DM, cloud resources	0.855	0.277	0.855	0.109
P014	NLDO-DM, cloud resources	0.771	0.322	0.771	0.130
P015	NLDO-DM, cloud resources	0.789	0.326	0.789	0.139

Table 16: **Recorded-evaluator scores by episode.** Ten seeds, t00–t12, 200×200 maximum budget. Scalar rows give objective gap; Pareto rows give recorded HV, ideal gap, and IGD. Table 13 gives public-formula Pareto scores.

Method	Attempted updates	Feas.|attempt	Valid prefix	Mean accepted update count	DS quality	DM HV
Persistent ReAct	126/180	96.8%	58/180	3.87	0.469	0.024
ReAct	55/180	83.6%	31/180	2.07	0.287	0.000
OptiMUS	53/180	83.0%	31/180	2.07	0.231	0.056
ORLM	35/180	77.1%	5/180	0.33	0.046	0.000
OptimAI	18/180	61.1%	9/180	0.60	0.037	0.024
OR-LLM-Agent	26/180	65.4%	12/180	0.80	0.056	0.013

Table 17: **Update-only continuity and quality**, t01–t12. Feas.|attempt considers attempted updates. Mean accepted update count, $\mathrm{E}[\mathrm{horizon}]$, counts updates in the strict prefix. Prefix, mean count, and quality assign zero to failed suffixes. DM uses public formulas. MAPLE's DS quality and DM HV are 0.947 and 0.879. Table 3 includes initialization.

Episode	First break	Direct cause	Observable evidence
P007	t03	Missing output field	The code compiled and ran, but a required reasoning field was missing. Code-output repair did not check this field.
P008	t05	Missing output field	The code compiled and ran, but a required reasoning field was missing. Code-output repair did not check this field.
P009	t02	Missing output field	The code compiled and ran, but a required reasoning field was missing. Code-output repair did not check this field.

Table 18: **First protocol failures in three Persistent ReAct rostering sequences.** Each saved plan is mathematically feasible at the first rejected state, but omits a required reasoning field.

A.13 Component and Restart Evidence

Agreement between predicted and saved edits. We compare LPD's three edit decisions with the changes saved after each update. The saved decisions are those used at seed 0. All ten numerical seeds share the same LLM-generated data and Workbench. The analysis contains 180 language decisions. LPD predicts changes to data, decision definition, and evaluation. All three predictions match the accepted changes in 170 updates. The separate counts are 174, 178, and 178. The six data differences are explicit roster-table replacements at P007–P009 t11–t12. In the other four cases, LPD requests a cautious code edit but the model returns the function unchanged. Whenever a function changes, LPD has marked it for change.

Thirty benchmark updates refer to entities through earlier requests (Table 20). LSM resolves all 30, including six requests that require following three reference links across the update history.

	NLDO-DS		NLDO-DM	
Method	**Strict**	**Mathematical**	**Strict**	**Mathematical**
ReAct	33.3%	37.6%	2.6%	17.9%
Persistent ReAct	51.3%	85.5%	11.5%	42.3%
ORLM	12.0%	29.9%	0.0%	3.8%
OptiMUS	28.2%	33.3%	12.8%	21.8%
OR-LLM-Agent	11.1%	17.1%	12.8%	12.8%
OptimAI	7.7%	10.3%	11.5%	14.1%

Table 19: **Solve rate under two prefix rules.** The same saved trajectories cover t00–t12, with 117 scalar and 78 Pareto states per method. The mathematical prefix omits auxiliary output-field checks but retains feasibility and zero failed or missing suffixes. MAPLE solves 100.0% under both rules. Quality under these rules appears in Table 3.

Reference links	Updates	Entity/field	Value	Valid result
1	3	3	3	3
2	21	21	21	21
3	6	6	6	6

Table 20: **Reference grounding with stored events.** None of the 30 current requests states the target entity ID. Entity/field and value compare the accepted public-data change with the benchmark's intended update. The intended answer is never shown to the language model.

These requests test whether earlier update records identify the required entities. The following control also tests requests that need assignments from the accepted plan.

Information needed to interpret assignment requests. We vary whether the model receives earlier update records, the accepted plan, both, or neither. We use saved routing states from P010 and cloud-placement states from P015 at t04, t08, and t10. The requests require three kinds of information. One kind identifies the requested assignment through earlier updates. Another names the relevant entities and asks to preserve assignments from the accepted plan. The third uses an earlier update to identify the entities and the accepted plan to retrieve their assignments. Each kind has three equivalent wordings in each domain, giving 18 language requests.

For example, the first routing request assigns the customer identified in a congestion note to the vehicle identified in an emissions update. The second names orders O001, O002, and O003 and preserves their saved vehicle assignments. The third preserves the saved vehicle assignment of the customer identified in the congestion note. Table 21 contains the recorded request texts.

A fifth condition supplies the required assignment through a fixed procedure. The required answers are stored separately for evaluation.

Saved plans used in the language tests and search replay. The original language tests and the search replay use representatives selected by different rules. The language tests read a representative reconstructed from each seed's saved archive. They select the plan with the lowest recorded scalar fitness and break ties by archive order. The original online run selects its representative by a different rule. It stores the first member of the final NSGA-II population, ordered by nondomination rank, decreasing crowding distance, and violation score, with stable ties.

For the search replay, we use the saved query from the first wording and read assignments from the originally exported online plan. We run six cases with three search seeds under each of the five information conditions. This gives 18 numerical results per condition and 90 results in total. Each search uses a population of 200 and at most 200 generations. The public tables, Workbench, Warm choice, starting population, and search budget are held fixed across conditions. The requested assignment is imposed after the restart population is built.

Profile	Information needed	Public request q0
P010	Earlier update records	For the next dispatch window, the customer previously marked as the congestion anchor must be served by the vehicle whose telemetry update changed its emission rate. Treat this as a hard assignment constraint. All other feasibility rules and Pareto objectives stay unchanged.
P010	Accepted plan	For the next dispatch, keep the currently accepted vehicle assignments of orders `O001`, `O002`, and `O003`. Other orders may move, and the original feasibility rules and Pareto objectives remain unchanged.
P010	Both records	Preserve the currently accepted vehicle assignment of the customer referred to as the congestion anchor. Other orders may be reassigned, and the original objectives remain unchanged.
P015	Earlier update records	For the next placement window, assign the first job named when workload group `burst-2` was introduced to machine `M07`. This is a hard assignment constraint. All other feasibility rules and Pareto objectives stay unchanged.
P015	Accepted plan	For the next placement, keep the currently accepted machine assignments of jobs `J012`, `J013`, and `J014`. Other jobs may move, and the original feasibility rules and Pareto objectives remain unchanged.
P015	Both records	Keep the current accepted machine assignment of every job in the workload group called `burst-2`. Other jobs may move, and the original objectives remain unchanged.

Table 21: **Requests that use different stored information.** The released cases include all three wordings and identifiers for their source states.

Identifying and enforcing the requested assignment. We evaluate assignment identification in the language tests and assignment enforcement in a separate search replay. Full LSM resolves all 18 requests. With only earlier update records, the model resolves the six requests answerable from those records. With only the accepted plan, it resolves the six requests that explicitly name the relevant entities. Neither condition resolves the six requests that need both types of information. With neither record, no request is resolved. The supplied-assignment condition returns the required relation directly.

The numerical test requires the correct assignment to be retrieved and enforced. Every plan in the returned archive must satisfy the problem constraints and preserve that assignment. Missing or incorrect assignments and infeasible archives receive zero quality. Full LSM and the supplied-assignment control each produce 18 valid results and mean HV of 0.825. The earlier-records-only condition produces six valid results and mean HV of 0.261. The accepted-plan-only condition produces six valid results and mean HV of 0.258. Table 22 separates these numerical results from the original language tests.

Comparison of reconstructed and online plans. We compare the requested assignments in the reconstructed representatives with those in the original online plans. The reconstructed representative and the original online plan differ in 12 of the 18 case–seed states. Among the twelve cases that read an accepted plan, the requested assignments differ in five. The differences occur in the requests that name orders and preserve their assignments in P010 (seeds 0 and 1), and in the requests that use both records in P015 (seeds 0–2). The requested assignments agree in the other seven cases. The six requests answered from earlier update records do not read the accepted plan.

The five conditions share the same starting population for each case and seed.

Program construction without TSS. Without TSS, the model defines its own candidate representation and the functions that generate, evaluate, modify, and repair candidates. It retains this program through the sequence. At `t00`, the model receives the public problem and the required Workbench functions. It receives no TSS decision types, encoding guidance, or type-specific search operators. It also defines data loading and conversion from a candidate to a solution. Under Warm, it can repair only solutions from its own preceding state. LPD still identifies which part of the problem changed. LSM still supplies stored events, the method's own accepted state, and its search archives. The fixed optimization loop still summarizes the change, applies the same restart selector, performs selection,

(a) Identifying the requested assignment: original language tests

Information provided	Uses earlier updates	Uses the accepted plan	Uses both records
Full LSM	6/6	6/6	6/6
Earlier update records only	6/6	0/6	0/6
Accepted plan only	0/6	6/6	0/6
Current request only	0/6	0/6	0/6
Assignment supplied directly	6/6	6/6	6/6

(b) Preserving the assignment during search: replay results

Information provided	Valid search results	Mean HV
Full LSM	18/18	0.825
Earlier update records only	6/18	0.261
Accepted plan only	6/18	0.258
Current request only	0/18	0.000
Assignment supplied directly	18/18	0.825

Table 22: **Using stored events and the accepted state.** (a) Original grounding results across three wordings, using minimum-recorded-fitness archive representatives. (b) Replay of the saved q0 queries against original online plans: six cases and three numerical seeds per condition, scored by the recorded evaluator. Missing or incorrect assignments and infeasible archives receive zero. In the supplied-assignment condition, a fixed procedure provides the expected relation directly.

maintains the archive, checks stopping, and formats the output. Compilation, public validation, and the repair limit are unchanged. This control removes the typed representation, type-specific operators, and restricted editing interface together.

One generated Workbench sequence is used per episode. Seeds 0–9 evolve independent populations under that fixed model run. Across P010–P015, the six model runs contain 13 distinct Workbench versions. All 13 saved Workbench versions compile. The failing routing version omits vehicle uniqueness, producing 120 invalid update–seed outputs. The failing cloud version records GPU incompatibility without clearing the feasibility flag, producing 20 invalid outputs. The held-out evaluator assigns zero quality to these outputs.

Episode	Failed results	Observed violation	Missing check
P010 (routing)	120/120	one vehicle assigned to two routes	vehicle uniqueness omitted from the evaluation function
P015 (cloud)	20/120	GPU-incompatible placements	violations recorded without clearing the feasible flag

Table 23: **Hidden constraint failures of the persistent untyped Workbench.** Counts use the hidden evaluator on the final result of every failed update–seed result. The Workbench reports all 140 results as feasible. The hidden evaluator finds a hard-constraint violation, so each result contributes zero to online quality.

Combining and modifying candidates. This test holds the optimization program, starting candidates, repair procedure, and restart choice fixed. Table 24 reports the results. At every P010/P015 update, both methods receive the same MAPLE population, Workbench, restart action, typed repair, and consistency checks. The control replaces search operators designed for permutations, assignments, and values. It instead selects a complete decision segment from one parent or samples a new valid segment at random.

Variation	Solve rate	P010 HV	P015 HV	All HV
Typed operators	100.0%	0.756	0.792	0.774
Generic random sampling	91.7%	0.657	0.579	0.618

Table 24: **Comparison of type-specific search operators.** Recorded-evaluator values cover t01–t12 with numerical seeds 0–2 under the same 200×200 limit and held-out reference. Only crossover and mutation change.

Variation across independently generated programs. We generate three program sequences for MAPLE and its w/o-TSS variant on P010 routing and P015 cloud placement. The three sequences include the original main sequence. Each sequence uses search seeds 0–2 across all twelve updates. The Warm–Full choices, evaluation functions from the original runs, 200×200 budget, archive limit of 500, and stopping rule remain fixed. Final scores use the same held-out $500 \times 500 \times 10$ reference. A model run that does not produce a valid Workbench receives zero solve rate and online quality for its requested results.

Method	Model run	Execution completed	Solve rate (%)	Online quality	Quality if valid	Numerical SD
P010 (routing)						
MAPLE	0	yes	100.0	0.756	0.756	0.120
	1	yes	100.0	0.693	0.693	0.013
	2	yes	100.0	0.560	0.560	0.038
	All	3/3	100.0 ±0.0	0.669 ±0.100	0.669	0.057
MAPLE w/o TSS	0	yes	0.0	0.000	–	0.000
	1	yes	100.0	0.435	0.435	0.016
	2	yes	100.0	0.572	0.572	0.029
	All	3/3	66.7 ±57.7	0.336 ±0.299	0.503	0.015
P015 (cloud)						
MAPLE	0	yes	100.0	0.792	0.792	0.022
	1	yes	83.3	0.708	0.850	0.004
	2	yes	83.3	0.685	0.822	0.010
	All	3/3	88.9 ±9.6	0.728 ±0.056	0.819	0.012
MAPLE w/o TSS	0	yes	83.3	0.751	0.901	0.011
	1	yes	100.0	0.059	0.059	0.001
	2	yes	83.3	0.729	0.874	0.019
	All	3/3	88.9 ±9.6	0.513 ±0.393	0.577	0.011

Table 25: **Independent model runs on one routing and one cloud episode.** Recorded-evaluator control. Each run independently generates a Workbench sequence. Numerical seeds 0–2 then search that sequence over t01–t12. Warm–Full choices remain fixed, so only Workbench construction and maintenance vary. Solve rate uses the held-out validity check. Online quality sets invalid results to zero. The *All* rows report mean ± sample standard deviation across model runs. Quality if valid includes accepted states only. Numerical SD measures variation across search seeds within one model run.

Across three Workbench runs, MAPLE's mean online quality is 0.669 in routing and 0.728 in cloud, compared with 0.336 and 0.513 without TSS. All three typed routing runs remain valid. The two additional typed cloud runs fail the held-out check at t11–t12. Table 25 reports model-run variation separately from numerical-seed variation for these two episodes.

Evaluation with a second language model. We repeat the full benchmark with Kimi k2.7 using the same evaluation and search settings. Table 26 reports the results under the common public-formula evaluation protocol.

	DS		DM	
Language model	**Solve rate**	**Online quality**	**Solve rate**	**Online quality**
MAPLE (DeepSeek-V4-Pro Preview)	100.0%	0.951	100.0%	0.875
MAPLE (Kimi k2.7)	100.0%	0.937	92.3%	0.766

Table 26: **Full-benchmark MAPLE replication across language models.** Both rows use the complete MAPLE system with one model run per episode and ten numerical seeds. Public-formula evaluation, search budget, and stopping rule match Table 3.

Model calls and tokens for program construction. We group recorded calls into program construction, identification of requested edits, and input-table updates. The grouping uses each recorded call's system prompt. Tables 27–28 report means over four additional runs per method on P010/P015. Each run covers twelve updates with three search seeds and fixed restart choices. Code construction includes generation, modification, and repair. Token counts include both input and output. The w/o-TSS runs reuse saved localization decisions and data patches.

Method	**Code calls**	**Code tokens**	**Tokens for identifying requested edits**	**Tokens for updating input tables**
MAPLE	4.0	51.6k	68.9k	135.0k
MAPLE w/o TSS	4.8	72.8k	Reused	Reused

Table 27: **Code construction uses fewer tokens with TSS.** Reused denotes update processing supplied from existing records.

Initial code generation averages 19.1k tokens with TSS and 26.0k without it. Code modification averages 32.5k and 46.9k, respectively, including repair calls in both categories. The combined reduction is computed before rounding.

Method	**Recorded scope**	**Tokens**	**Calls**	**Provider min.**	**Wall min.**
MAPLE	Code, localization, data patches	255.4k	27.8	14.7	19.9
w/o TSS	Code; supplied update processing	72.8k	4.8	6.5	11.1

Table 28: **Execution metadata for the same runs as Table 27.** Wall time includes numerical search.

Mean input/output counts are 202.6k/52.9k for MAPLE and 47.7k/25.1k for w/o TSS. Input cache hits of 41.4k and 6.7k are included in the input totals.

Reuse strategies over complete update sequences. Each strategy starts from the same initial state and continues with candidates produced by its own searches. The initial state is the typed Workbench and population at t00. Each strategy proceeds through t01–t12. Always Full keeps the TSS Workbench but draws all 200 candidates anew at every update. The policies use the same saved Workbench at each state: 60 initial execution instances and 720 update execution instances across six episodes and ten numerical seeds.

Across six episodes, the mean paired gain over Always Warm is 0.010. Five episode means are positive. The episode-cluster bootstrap interval is [0.001, 0.021], and the two-sided exact sign test gives $p = 0.219$. The gain is concentrated in routing. The mean gain over Always Full is 0.054. Table 12 reports the results by domain.

Recorded reuse and restart decisions. We report the initialization action selected before search for each saved update.

Across P007–P015, the selector chooses Warm at t01–t10 for all nine episodes. It chooses Full at t11 for all nine. At t12, it chooses Full for P008 and P010–P015 and Warm for P007/P009. The total is 16 Full and 92 Warm decisions. These decisions are made before numerical optimization

Sequential policy	Full restarts	Solve rate	Online quality	Generations	Δ quality vs. MAPLE
MAPLE (ours)	12/72	100.0%	0.879	162.8	—
Always Warm	0/72	100.0%	0.868	162.8	−0.010
Always Full	72/72	100.0%	0.824	199.0	−0.054

Table 29: **Restart policies that follow their own populations on P010–P015.** Public-formula evaluation of t01–t12. Warm and Full carry their own results through all updates. Every policy uses the same TSS Workbenches, 200×200 maximum budget, 10 seeds, and stopping rule. Generations are taken from the original executions. Paired differences are computed before rounding. Displayed means are rounded separately.

and are identical across the 10 numerical seeds. The disruptive updates appear at t11–t12 by design. At P010–P012 t08, the selector chooses Warm and cites the retained decision representation, active entities, and objective declarations. On the main Pareto trajectories, the deterministic table-change rule makes the same Warm–Full choices as the selector.

Restart decisions based on changed input cells. We compare MAPLE with a rule that starts afresh when at least half of the counted input cells change. For every table with an `active` field, it counts the cells in rows whose current `active` value is true, excluding the `id` and `active` fields themselves. This count is the table's denominator. Rows are matched by their raw public `id` values. A row with an identifier absent from the preceding table, or previously inactive, contributes all its counted cells to the numerator. For other rows, only unequal cell values count as changed. The rule uses the maximum table fraction, with a zero fraction for an empty denominator. The rule chooses Full when this fraction is at least 0.5 and Warm otherwise. Its choices are fixed before the Warm and Full results are examined. Across P010–P015, the ratio ranges from 0 to 0.060 on ordinary updates and from 0.705 to 1.000 at t11–t12. The rule therefore matches all 12 Full decisions from the restart selector. It obtains the same 0.779 HV under the recorded evaluator and 162.8 mean generations without a model call or a new optimizer run.

Reuse after rescaling, renaming, and task addition. Starting from saved routing and cloud-placement states, we rescale values, rename entities, or add small tasks. Each additional test starts at t05 of P010 or P015. Each affects more than half of the counted table cells. The editable Workbench functions, decision types, and required output remain unchanged, while the number of active decisions grows when tasks are added. A fixed procedure constructs the tables and identifier mappings without model calls. The ID-renaming cases retain the same table and field names.

Rescaling routing parameters. For routing rescaling, depot and order coordinates, order demand, ready/due/service times, vehicle capacity, and shift end are multiplied by 10. Priorities and emission rates stay fixed. The historical Workbench identifies congestion by the original coordinate pair $(48, 11)$ (Figure 6). After coordinate rescaling, that match disappears, so this branch changes the incident's effect as well as the parameter scale. Its table-change ratio is 0.86.

Rescaling cloud parameters. For cloud rescaling, job CPU and memory demands, machine CPU and memory capacities, and idle energy are multiplied by 10. Job deadlines, latency sensitivities, and priorities are also multiplied by 10 but enter only diagnostics in this Workbench. GPU requirements, availability, per-CPU energy rates, and global energy multipliers stay fixed. For any fixed feasible assignment, utilization and the recorded imbalance B are unchanged, while energy becomes $E' = 10E$. Thus feasibility and Pareto dominance are preserved, although scalar Warm repair uses the changed weighting $10E + B$. The table-change ratio is 0.71.

Renaming entities. Relabeling consistently renames orders and vehicles, or jobs and machines, without changing their other attributes (ratio 1.00). The table-change rule does not apply the identifier mapping before matching rows, so every renamed row is treated as new even though its non-identifier values are retained.

Adding tasks. Task additions introduce 20 low-demand orders near the depot or 26 small CPU/memory jobs with loose deadlines (ratios 0.53 and 0.52). These additions enlarge the task while retaining the existing resources and earlier demands.

Comparing the restart choices. Warm and Full begin with the same saved candidate population and use three search seeds. They use the original objective definitions and the combined nondominated reference for each additional test. The table-change rule and the numerical rule that evaluates candidates before updating their identifiers choose Full on all six cases. The restart selector chooses Warm on all six and cites the rescaling, renaming, or easily placed additions in its recorded assessment.

Warm obtains mean HV of 0.862 across the six cases, compared with 0.682 for Full. The largest difference is +0.708 for Warm (Table 30). The six cases cover routing and cloud placement. The routing-rescaling case also changes how congestion is applied. P015 recovers almost fully from fresh restarts, so most of the quality difference comes from routing. Warm uses fewer generations in five of the six branches. Table 5 presents the relabeling and task-addition results compactly alongside the disruptive main-sequence updates. The rescaling branches and the pure-scaling numerical control below are supplementary diagnostics.

Case	Warm HV	Full HV	Paired Δ HV Mean $\pm$ SD	Min. Δ HV	Max. Δ HV
P010 parameter rescaling	0.716	0.510	$+0.206 \pm 0.433$	-0.294	+0.478
P010 ID relabel	0.752	0.603	$+0.149 \pm 0.196$	-0.060	+0.329
P010: Added low-demand orders	0.769	0.061	$+0.708 \pm 0.236$	+0.442	+0.892
P015 parameter rescaling	0.989	0.990	-0.002 ± 0.003	-0.005	+0.001
P015 ID relabel	0.980	0.980	0.000 ± 0.012	-0.013	+0.008
P015: Added small jobs	0.968	0.946	$+0.022 \pm 0.021$	+0.005	+0.045
All six cases	0.862	0.682	+0.181	–	–

Table 30: **Reuse after rescaling, relabeling, and task additions.** Six t05 branches, three paired seeds per action, and 36 executions. Paired differences $\Delta_s = \mathrm{HV}_{\mathrm{Warm},s} - \mathrm{HV}_{\mathrm{Full},s}$ use the same incoming population and each branch's pooled reference under the recorded evaluator. SD is the sample standard deviation, and min./max. are observed seed extrema. The last row averages branch means. Routing rescaling also removes the coordinate-based congestion match.

For each branch, $\bar{\Delta} = \frac{1}{3}\sum_{s=0}^{2} \Delta_s$ and $s_\Delta = \sqrt{\frac{1}{2}\sum_{s=0}^{2}(\Delta_s - \bar{\Delta})^2}$. The three routing task-addition differences are all positive, ranging from 0.442 to 0.892. Routing rescaling and relabeling have positive means but each includes a negative seed difference. The cloud differences are small, consistent with the similar final quality and earlier Warm stopping discussed above.

Updating candidate identifiers before evaluation. We replace the identifiers in each old candidate before evaluating it against the renamed tables. For relabeling, we compare $f_{\mathrm{old}}(x)$ with $f_{\mathrm{new}}(T(x))$, where T applies the public ID mapping to assignment keys, assigned resources, and route permutations. The numerical selector retains its original sample indices and thresholds. All 1,200 incoming candidates across the two profiles and three seeds preserve their objective vectors and feasibility after mapping. The six sampled decisions change from Full to Warm: both objective displacement and feasibility loss become zero.

Reuse under uniform routing rescaling. We keep the congestion rule attached to the same order before and after scaling, then compare Warm and Full. We replace the coordinate lookup with the same public order ID in both pre-update and scaled Workbenches. The incident's historical effect and all other recorded scoring rules remain fixed. On all 600 incoming candidates, the unscaled rewrite reproduces the historical evaluator, and scaling multiplies each objective by 10 while preserving feasibility. The numerical selector has $r = 0.9$ and chooses Full for all three seeds.

Warm has higher HV for every routing seed in these controls. The cloud difference is small: the three paired gains are 0.0054, -0.0023, and 0.0015. In the pure-scaling case, Warm uses 97.7 generations on average and Full uses 200. The generation counts do not include the extra evaluations used to adapt and repair old candidates.

The 18 restart-and-search calls take 170.6 seconds in total, including migration and repair but excluding source-state reconstruction and scoring.

Additional control	Warm HV	Full HV	Paired Δ HV
P010: Candidates updated to renamed IDs	0.786	0.395	$+0.391 \pm 0.278$
P015: Candidates updated to renamed IDs	0.984	0.982	$+0.002 \pm 0.004$
P010 pure scaling	0.791	0.466	$+0.325 \pm 0.299$

Table 31: **Reuse with consistent representations.** Eighteen new searches: three branches, three paired seeds, and two actions, with population 200 and a 200-generation cap. Each branch uses the pooled nondominated reference from its six searches. Differences are Warm minus Full, reported as mean $\pm$ sample SD.

Control	Seed	Warm HV	Full HV	Δ HV	Generations
P010: Candidates updated to renamed IDs	0	0.848	0.310	$+0.538$	75/200
	1	0.966	0.402	$+0.564$	70/200
	2	0.545	0.474	$+0.071$	158/200
P015: Candidates updated to renamed IDs	0	0.983	0.977	$+0.005$	176/200
	1	0.983	0.986	-0.002	200/200
	2	0.985	0.984	$+0.002$	200/200
P010 pure scaling	0	0.840	0.649	$+0.191$	65/200
	1	0.971	0.303	$+0.668$	70/200
	2	0.562	0.446	$+0.116$	158/200

Table 32: **Individual seeds for the offline controls.** Generations are Warm/Full. Each pair starts from the same incoming population.

Restart actions from a shared candidate population. For each update, both restart actions receive MAPLE's same saved candidate population. These are the Matched Warm and Matched Full controls. Only MAPLE's selected result advances the continuing trajectory. All action-matched comparisons use the recorded evaluator with population 200, a 200-generation cap, seeds 0–9, the same stopping rule, and the same held-out $500\times500\times10$ reference for final scoring. The Warm/Full comparison contains 1,080 paired episode–time-step–seed results per method over P007–P015. The numerical selector covers the 720 Pareto results on P010–P015.

Policy	Full restarts	HV	Generations	Gap to best action
MAPLE	120/720	0.779	162.8	0.004
Matched Warm	0/720	0.766	162.7	0.017
Numerical selector	245/720	0.765	169.4	0.018
Matched Full	720/720	0.731	199.0	0.053
Best action after results	300/720	0.783	167.8	0.000

Table 33: **Action-matched restart under the recorded evaluator.** All policies solve every state.

Re-evaluating old candidates to select a restart action. We evaluate a sample of earlier candidates under the old and revised problems before selecting Warm or Full. This comparison follows a common response to change in dynamic evolutionary optimization (Deb et al., 2007; Liu et al., 2014; Sahmoud & Topcuoglu, 2019). For each seed and update, it samples 20 of the 200 incoming candidates without replacement. It evaluates the same candidates with the old and updated Workbenches. For objective j, it computes

$$r_j = \frac{1}{K}\sum_{i=1}^{K}\frac{|f_j(x_i,t) - f_j(x_i,t-1)|}{\max(|f_j(x_i,t)|, |f_j(x_i,t-1)|, 10^{-12})}, \qquad r = \max_j r_j.$$

K is the number of retained valid objective pairs. It chooses Full when $r \geq 0.5$. It also chooses Full when at least half of the previously feasible sampled candidates become infeasible, the objective dimension changes, or no valid objective pair remains. A valid pair contains finite objective vectors of

the same nonzero length. If the old evaluation fails, the rule uses the saved fitness. Other unsuccessful pairs are omitted. It chooses Warm otherwise. The maximum over objectives keeps one large change from being averaged away. The relative denominator makes objective units comparable. Sampling is fixed by the numerical seed and time step. The numerical selector evaluates each unchanged incoming genome under both Workbenches before migration or ID mapping. All six relabeling seed cases retain 20 defined objective pairs, but lose feasibility for all sampled candidates and have $r = 1$. Their Full decisions therefore include representation mismatch. The mapped-candidate control in Appendix A.13 removes this mismatch and changes all six decisions to Warm.

Episode	Full t01–t10	Full t11–t12	Numerical HV	MAPLE HV	Matched Warm HV
P010	23/100	20/20	0.729	0.752	0.724
P011	24/100	20/20	0.710	0.736	0.712
P012	28/100	20/20	0.745	0.785	0.764
P013	10/100	20/20	0.855	0.854	0.853
P014	20/100	20/20	0.768	0.763	0.761
P015	20/100	20/20	0.785	0.785	0.784
All	125/600	120/120	0.765	0.779	0.766

Table 34: **Numerical change as a restart signal.** Recorded-evaluator, action-matched control over t01–t12. Counts are episode–time-step–seed results. The numerical selector chooses Full on all 120 disruptive update–seed cases and on 125 of the 600 ordinary cases.

Appendix A.15 reports the threshold comparison.

Variant	DS Quality	DS Gen.	DM HV	t11 HV	t12 HV	DM Gen.
MAPLE	0.927	66.7	0.779	0.561	0.426	162.8
Matched Warm	0.927	66.8	0.766	0.509	0.327	162.7
Matched Full	0.919	71.1	0.731	0.561	0.426	199.0

Table 35: Recorded-evaluator, action-matched control over t01–t12. **Restart actions from the same incoming population.** All methods solve all 360 DS and 720 DM time-step–seed results. Quality is normalized scalar score for DS and held-out-reference HV ratio for DM. Gen. is the mean generation count. The t11/t12 columns show MAPLE's 0.053/0.099 gains over Matched Warm.

Table 35 reports stage-end solution quality and the number of evolutionary generations for each restart policy. On P010–P015, MAPLE improves over Matched Warm by 0.013 normalized HV (episode-cluster 95% bootstrap interval $[0.004, 0.022]$), with the gain concentrated at t11–t12. All six episode means favor MAPLE over Matched Warm (two-sided exact sign test, $p = 0.031$). MAPLE recovers 74% of the improvement obtained by choosing the better action after seeing both outcomes. The mean HV gain over Matched Full is 0.048 (hierarchical paired 95% CI $[0.005, 0.097]$). MAPLE uses 162.8 generations on average, compared with 199.0 for Matched Full. At t11–t12, where the selector chooses Full for all six Pareto episodes, it matches the corresponding Matched Full controls. On P007–P009, MAPLE and Matched Warm have nearly the same result. Always using Full is slightly worse and uses more generations.

The per-episode breakdown in Table 37 localizes this effect. Across routing episodes P010–P012, the disruptive-update gain over Matched Warm ranges from 0.122 to 0.168 HV. The corresponding cloud gains are 0.002–0.011, consistent with the stronger benefit of fresh initialization in routing.

Routing Family mean: **M 0.757**, **N** 0.391, **W** 0.733, **F** 0.656

Time	P010				P011				P012			
	M	**N**	**W**	**F**	**M**	**N**	**W**	**F**	**M**	**N**	**W**	**F**
All-t	**0.752**	0.000	0.724	0.663	**0.736**	0.327	0.712	0.662	0.785	**0.845**	0.764	0.643
t01	**0.746**	0.000	**0.746**	0.710	**0.731**	0.379	**0.731**	0.663	0.847	**0.995**	0.847	0.684
t02	**0.774**	0.000	**0.774**	0.657	**0.776**	0.492	**0.776**	0.726	**0.872**	0.838	**0.872**	0.744
t03	**0.791**	0.000	**0.791**	0.769	**0.778**	0.440	**0.778**	0.730	0.815	**0.880**	0.815	0.642
t04	**0.778**	0.000	**0.778**	0.696	**0.768**	0.450	**0.768**	0.704	0.832	**0.931**	0.832	0.659
t05	**0.805**	0.000	**0.805**	0.668	**0.797**	0.397	**0.797**	0.692	0.838	**0.869**	0.838	0.640
t06	**0.849**	0.000	**0.849**	0.751	**0.842**	0.241	**0.842**	0.772	0.872	**0.895**	0.872	0.723
t07	**0.850**	0.000	**0.850**	0.733	**0.846**	0.289	**0.846**	0.736	0.876	**0.940**	0.876	0.688
t08	**0.862**	0.000	**0.862**	0.764	**0.784**	0.463	**0.784**	0.629	0.905	**0.931**	0.905	0.699
t09	**0.886**	0.000	**0.886**	0.659	**0.804**	0.532	**0.804**	0.715	0.891	**0.938**	0.891	0.706
t10	**0.910**	0.000	**0.910**	0.785	**0.918**	0.238	**0.918**	0.797	**0.898**	0.797	**0.898**	0.766
t11	**0.342**	0.000	0.230	**0.342**	**0.239**	0.000	0.040	**0.239**	0.397	**0.656**	0.395	0.397
t12	**0.427**	0.000	0.203	**0.427**	**0.545**	0.000	0.458	**0.545**	0.370	**0.469**	0.129	0.370

Cloud Family mean: **M** 0.801, **N 0.821**, **W** 0.799, **F** 0.805

Time	P013				P014				P015			
	M	**N**	**W**	**F**	**M**	**N**	**W**	**F**	**M**	**N**	**W**	**F**
All-t	**0.854**	0.837	0.853	0.851	0.763	**0.870**	0.761	0.773	0.785	0.755	0.784	**0.791**
t01	**0.915**	0.813	**0.915**	0.873	0.815	**0.905**	0.815	0.833	0.826	**0.883**	0.826	0.816
t02	0.917	0.901	0.917	**0.929**	0.743	**0.880**	0.743	0.763	0.821	**0.896**	0.821	0.836
t03	0.899	0.874	0.899	**0.901**	0.782	**0.881**	0.782	0.764	0.846	**0.915**	0.846	0.859
t04	0.879	0.878	0.879	**0.898**	0.770	**0.892**	0.770	0.795	0.833	**0.900**	0.833	0.835
t05	0.911	0.902	0.911	**0.913**	0.823	**0.927**	0.823	0.840	0.834	**0.905**	0.834	0.854
t06	0.898	0.888	0.898	**0.909**	0.843	**0.927**	0.843	0.860	0.846	**0.908**	0.846	0.859
t07	0.924	0.914	0.924	**0.924**	0.748	**0.907**	0.748	0.786	0.813	**0.900**	0.813	0.840
t08	**0.887**	0.880	**0.887**	0.885	0.749	**0.897**	0.749	0.781	0.845	**0.915**	0.845	0.837
t09	**0.882**	0.866	**0.882**	0.873	0.751	**0.921**	0.751	0.725	0.834	**0.912**	0.834	0.831
t10	**0.891**	0.866	**0.891**	0.870	0.823	**0.921**	0.823	0.825	0.871	**0.920**	0.871	0.867
t11	0.768	**0.780**	0.763	0.768	0.762	**0.770**	0.767	0.762	**0.858**	0.000	0.857	**0.858**
t12	0.473	**0.486**	0.475	0.473	0.546	**0.617**	0.520	0.546	**0.197**	0.000	0.178	**0.197**

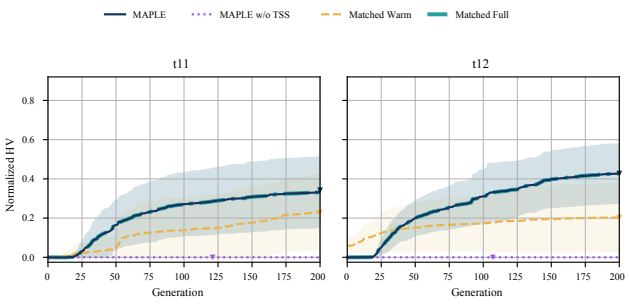
Table 36: **Recorded-evaluator results at each Pareto update.** M denotes MAPLE and N its independently generated untyped Workbench control. W and F denote Matched Warm and Matched Full, each initialized from MAPLE's incoming population. Values are ten-seed HV means, with invalid outputs set to zero. All-t averages t01–t12. Family means pool routing or cloud results.

Episode	All t01–t12			Disruptive updates t11–t12		
	MAPLE	Matched Warm	Matched Full	MAPLE	Matched Warm	Matched Full
P010	0.752	0.724	0.663	0.385	0.217	0.385
P011	0.736	0.712	0.662	0.392	0.249	0.392
P012	0.785	0.764	0.643	0.384	0.262	0.384
P013	0.854	0.853	0.851	0.621	0.619	0.621
P014	0.763	0.761	0.773	0.654	0.644	0.654
P015	0.785	0.784	0.791	0.528	0.517	0.528

Table 37: **Online quality for the six Pareto episodes.** Recorded-evaluator, action-matched control. All three policies have 100% solve rate. Each value is the normalized-HV mean over 10 seeds after infeasible results are set to zero. The right block isolates the disruptive t11–t12 updates.

Scalar restart policy	Full restarts	Online quality	Generations	Gap to best action
MAPLE	4/36	0.927	66.7	0.016
Matched Warm	0/36	0.927	66.8	0.016
Matched Full	36/36	0.919	71.1	0.024
Best action after results	6/36	0.943	66.5	0.000

Table 38: **Restart results on P007–P009.** Recorded-evaluator, action-matched t01–t12. All policies solve every state. MAPLE and Matched Warm both reach mean scalar quality 0.927. Gap uses the post-hoc best action.



Disruptive update. At t11–t12, the routing episode changes orders, depot, vehicles, and operating rules while retaining its representation and objectives.
Current action. The restart selector selects Full. It matches the Matched Full control because both use the same action and incoming state, and exceeds Matched Warm under this recorded evaluator.
Without TSS. The untyped Workbench assigns one vehicle to two routes, violating a hard constraint.

Figure 10: **Recorded within-search behavior at disruptive routing updates.** Ten-seed feasibility-gated HV means, standard-deviation bands, and mean stopping generations use the recorded reference. Figure 5 reports final quality under the fresh common public-formula references.

Curves remain constant after the marked stopping generation. Initial quality and recovery time differ: at t11–t12, Full can start below Warm and finish above it.

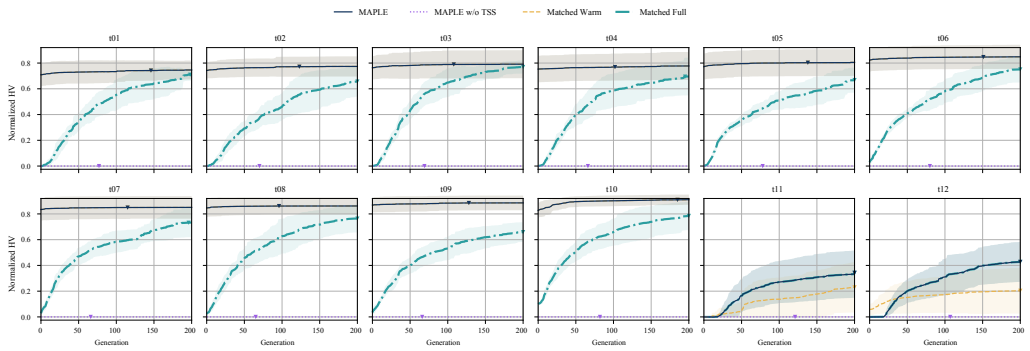


Figure 11: **Green-routing component results (NLDO-P010, t01–t12).** Recorded-evaluator mean HV over ten seeds, standard-deviation bands, and mean stopping generations (triangles). MAPLE and Matched Warm/Full remain feasible. Without TSS, duplicate vehicle assignment makes every P010 update infeasible with zero quality.

A.14 Static Solving Results

Tables 39 and 40 report the supplementary diagnostics defined in Appendix A.7.

Method	Output pass	Compile	Run	Objective	Named match	Joint Match
MAPLE static	33.9%	93.2%	93.2%	59.3%	37.3%	59.3%
Persistent ReAct	37.3%	100.0%	98.3%	62.7%	44.1%	62.7%
ReAct	27.1%	91.5%	78.0%	57.6%	32.2%	–
OptiMUS workflow	**69.5%**	96.6%	93.2%	**69.5%**	–	69.5%
ORLM workflow	23.7%	84.7%	61.0%	23.7%	–	23.7%
OptimAI	47.5%	**100.0%**	**98.3%**	61.0%	49.2%	–
OR-LLM-Agent workflow	50.8%	96.6%	96.6%	50.8%	–	50.8%

Table 39: **NLP4LP-hard diagnostic checks.** Output pass records each workflow's original acceptance flag. Named-variable agreement is reported separately. Joint Match requires execution and reference-objective/status agreement.

Method	Output pass	Compile	Run	Objective	Named match	Joint Match
MAPLE static	**80.0%**	**100.0%**	**100.0%**	**80.0%**	n.a.	80.0%
Persistent ReAct	70.0%	**100.0%**	**100.0%**	70.0%	n.a.	70.0%
ReAct	75.0%	**100.0%**	95.0%	70.0%	n.a.	–
OptiMUS workflow	65.0%	**100.0%**	**100.0%**	65.0%	n.a.	65.0%
ORLM workflow	45.0%	90.0%	55.0%	45.0%	n.a.	45.0%
OptimAI	60.0%	80.0%	80.0%	55.0%	n.a.	–
OR-LLM-Agent workflow	65.0%	95.0%	95.0%	65.0%	n.a.	65.0%

Table 40: **BWOR20 diagnostic checks.** Output pass, numeric objective agreement, and joint Match are distinct recorded criteria. Named-variable matching does not apply.

The two static benchmarks expose different weaknesses. On BWOR20, all 20 MAPLE rows compile and run, and 16 match the reference objective. On NLP4LP-hard, 55 of 59 rows compile and run, 35 match the objective, and 20 pass the additional historical output check.

A.15 Reference and Archive Sensitivity

Sensitivity to the reference set and evaluation box. We vary the reference set and evaluation box and recompute the reported HV ratios.

Table 41 examines two choices that remain when the Pareto reference is finite. For reference subsampling, each method uses the same fixed subset in each of 20 repeats. The evaluation box from the full reference remains fixed. For box sensitivity, both numerator and denominator are recomputed after expanding the ideal-to-reference span.

Under the recorded evaluator, five w/o-TSS outputs at P012-t01 exceed HV ratio 1. Expanding the HV box reduces the number of ratios above 1 from five to four and then one. The 500-member archive cap is inactive for the MAPLE results in this analysis.

Effect of the Pareto archive limit. We repeat the search with archive limits of 100, 200, and 500 while keeping the other settings fixed.

Caps 100 and 200 truncate the archive in 240 and 162 results, respectively, while cap 500 is inactive. Their mean HV ratios are 0.775, 0.776, and 0.774. The largest change in the mean is 0.0016. Mean IGD is 0.139, 0.139, and 0.138. The smaller limits do change individual fronts. Their mean absolute HV differences from limit 500 are 0.053 and 0.040. They also use 5.3 and 2.8 more generations on average. The mean is stable, but archive size can still matter at an individual time step.

	Reference subsampling: ΔHV ratio			HV-box span: mean ratio		
Method	**75%**	**50%**	**25%**	$1.00\times$	$1.10\times$	$1.25\times$
MAPLE	+0.005	+0.013	+0.039	0.779	0.796	0.818
Always Warm	+0.005	+0.013	+0.039	0.769	0.786	0.808
Always Full	+0.005	+0.012	+0.036	0.731	0.752	0.779
MAPLE w/o TSS	+0.004	+0.010	+0.028	0.606	0.621	0.640

Table 41: **Sensitivity to the finite Pareto reference on P010–P015, t01–t12.** Recorded-evaluator results. Each method contributes 720 time-step–seed results. The Warm and Full rows follow their own populations through the sequence. Reference subsampling reports the change from the full-reference mean. The box columns recompute both volumes. Method ordering is unchanged in every setting. For MAPLE w/o TSS, the number of results above one changes from 5 to 4 to 1 as the box expands. The maximum changes from 1.020 to 1.014 to 1.008.

Cap	P014 HV	P015 HV	Mean HV	IGD	Gen.	Active/results
100	0.760	0.791	0.775	0.139	188.4	240/240
200	0.771	0.780	0.776	0.139	185.9	162/240
500	0.763	0.785	0.774	0.138	183.1	0/240

Table 42: **Sensitivity to the archive limit on P014/P015 under the recorded evaluator.** Each row contains the same 240 update–seed results. The restart choices, 200×200 search budget, stopping rule, and held-out reference remain fixed. Only the maximum archive size changes.

Sensitivity to restart thresholds. We vary the numerical-change threshold and identify the interval that leaves the table-change decisions unchanged. The numerical selector uses a change threshold of 0.5 in the reported comparison. For thresholds of 0.25, 0.50, and 0.75, the numerical selector obtains HV ratios of 0.755, 0.765, and 0.767, with Full rates of 48.8%, 34.0%, and 25.0%. All three HV ratios are below MAPLE's 0.779 under the recorded evaluator. At threshold 0.50 it uses 169.4 generations, compared with 162.7 for Matched Warm and 162.8 for MAPLE. The table-change threshold lies in a ten-fold gap. Ordinary updates score 0–0.060, while disruptive updates score 0.705–1.000. Any value in $[0.3, 0.7]$ therefore makes identical decisions. Tables 42 and 41 separately test the archive limit and Pareto reference.

Sensitivity to congestion and return-deadline checks. We separately recompute scores after changing the congestion scope and enforcing the return deadline. Under the recorded reference, extending congestion to adjacent departure/return legs changes routing HV from 0.7471 to 0.7466 without changing MAPLE's ordering relative to Warm/Full. Four saved candidate occurrences violate the return deadline. Enforcing this deadline leaves the primary aggregate HV unchanged.

A.16 Using MAPLE through an Interactive Interface

The interface lets users submit a planning request, inspect the solutions, and revise the same problem. The application is called MAPLE Harness. Each session retains the Workbench, candidate plans, objective traces, and restart actions across updates. Figure 12 shows a CNC scheduling session from the published walkthrough, separate from NLDO.

The interface sends planning requests to a separate service that runs MAPLE through the Model Context Protocol (MCP). Each request returns a job identifier, which the interface uses to display search progress. After the result is accepted, the service saves the updated session. Users can inspect the solutions and export them as CSV or JSON.

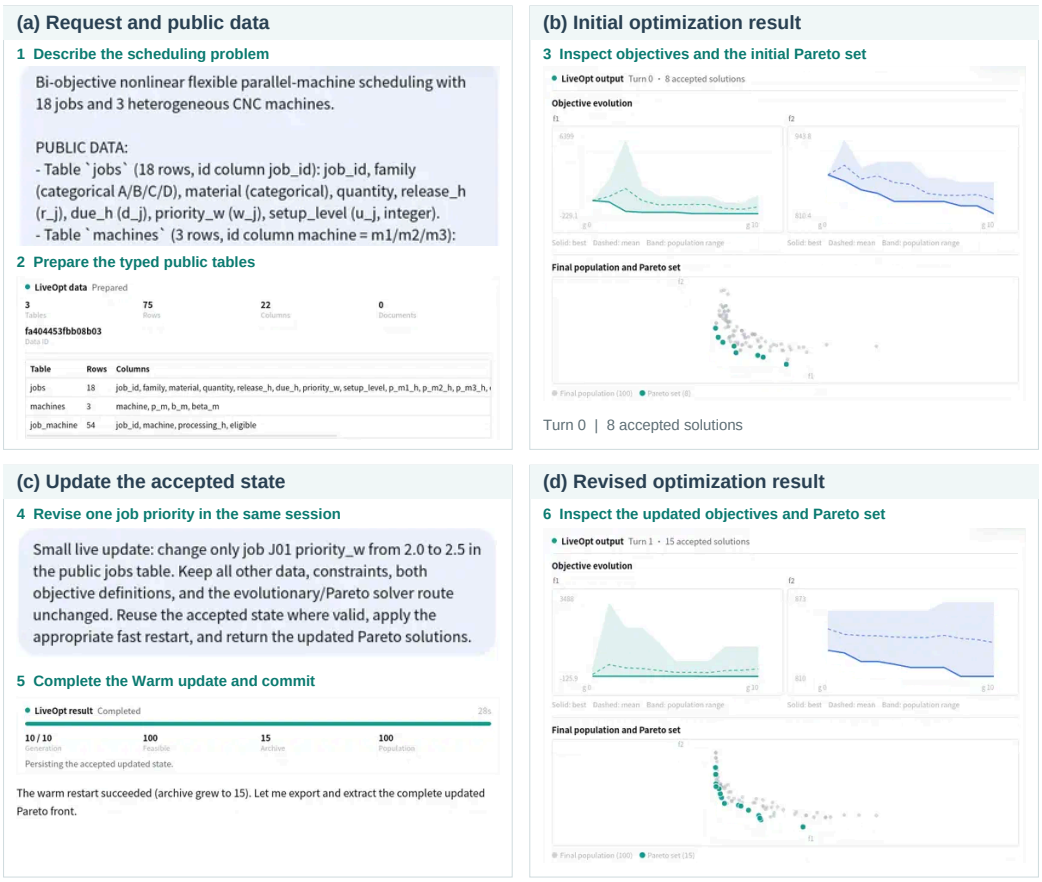


Figure 12: **A complete Harness interaction from request to accepted update.** Recorded steps: (a) scheduling request and public tables; (b) initial traces and eight Pareto solutions; (c) J01's priority changes from 2.0 to 2.5, followed by a Warm update; (d) revised traces and fifteen Pareto solutions.

A.17 Prompt Interfaces and Execution Rules

This section summarizes the recorded prompt interfaces, required output fields, and state-transfer rules. The TSS prompts specify two editable functions and the required table format. The other interfaces specify the functions used by search. Field names and response delimiters below retain their recorded spelling. Problem text, public tables, update history, previous output, and repair messages fill the variable fields. Table 10 defines the carried-state interfaces.

MAPLE TSS Workbench builder — initialization prompt

User task. The prompt shows a complete runnable Workbench, then asks the model to make the smallest necessary edits for the public problem. The two editable surfaces are `build_problem(public_context)` and `evaluate(genome, data)`.

Framework-owned code. Main loop, population initialization, GA/NSGA-II selection, LP/MILP solving, segment-specific crossover/mutation, nondominated sorting, crowding distance, archive tracking, and penalty-result construction.

Required response. Exactly two slot blocks. The decision-definition function receives fixed-code-normalized public tables, builds reusable data views, declares typed decision segments, chooses the solver mode, and lists objective names. The evaluation function evaluates one genome by decoding the candidate, computing public violations, scalar rank, diagnostics, and optional objective vector.

Rules. Do not generate a separate model schema, GA/NSGA code, a large data contract, or unused helper code. Do not hard-code entity ids, resource names, source ids, hidden references, or benchmark-specific constants. Loop over public table rows. Submitted solution identifiers must be scalar public IDs, not full row objects.

Public task definition. If `objective_mode=multi_objective`, the prompt requires `solver_mode="moea"` and objective names/order exactly matching the public task definition. Scalarization and literal constant Pareto dimensions are forbidden.

Repair turn. If compilation or smoke execution fails, the next prompt includes the previous public error plus the previous slot contents and asks for corrected Workbench slots.

MAPLE update localizer — dynamic classification prompt

System. `Classify the dynamic update for a fixed LiveOpt Workbench. Return exactly one JSON object.`

Input. Natural-language update, current segment signature, current public-context summary, representative accepted decision, public event/reference ledger, and allowed restart-skill names. The search population and archive are not shown.

Required JSON fields. `data_update`, `patch_setup`, `patch_fitness`, `restart_skill`, `reason`, and `state_binding_queries`.

Decision rules. Use `patch_setup` only when search segments or data extraction must change. Use `patch_fitness` when objectives, constraints, penalties, decode, or solution reporting must change. If public tables changed but the existing Workbench slots loop over `public_context`, prefer `data_update=true` and `patch_setup=false`.

State binding. For an explicit cross-stage assignment hold, return a typed `lock_assignment`, `preserve_demand_assignments`, or `preserve_resource_assignments` query and leave the code-patch flags false unless another part of the request changes the Workbench. Fixed code resolves accepted assignment values and checks them against the current typed segment.

Restart hint. A schema-required provisional identifier h_t retained for traceability. It is neither the final Warm–Full action nor an instruction to the Workbench editor. After the candidate Workbench passes public checks, the semantic selector and fixed gate replace it with the effective action α_t.

MAPLE semantic restart selector — reuse-risk assessment prompt

System. `Assess whether repaired solutions near the previous population are likely to remain competitive.`

Input. After editing, compilation, and the smoke test: an opaque update id, the public update text, and a fixed summary containing changed public field paths, representative row changes, old/new decision types, old/new public objective declarations, and changes to the editable functions. The selector also keeps the initial request and earlier accepted public updates in its conversation context.

Required JSON. `reuse_risk`, `change_mechanisms`, `full_vote`, `evidence_paths`, and `reason`. Evidence paths must exactly match the digest.

Decision rule. Warm already mixes at most 50% repaired history with fresh candidates. Full is reserved for a supported replacement of key decisions, reversal of objective preferences or resource roles, new constraint system, or risk that good solutions move far from the old population. Ordinary changes and uncertainty return Warm.

Boundary. The prompt explicitly forbids trial optimization, benchmark/time-step identity, difficulty labels, search-action advice in the event text, hidden references, HV, and IGD. The gate accepts Full when the

assessment reports high reuse risk, casts a Full vote, includes a structurally supported mechanism, and cites at least one changed field path.

MAPLE public-table updater — natural language to public table patch

System. `Patch public optimization data from a natural-language update. Return exactly one JSON object.`

Writable root. Only `public_context["tables"]`. Every table is a `list[dict]`. Singleton tables such as policy, settings, depot, or global parameters are still one-row lists.

Operations. `update_row`, `append_row`, `delete_row`, `set_value`, `replace_value`, and `replace_table`.

Required shape. A compact JSON object with `operations`, where each operation records `op`, `path`, optional `match`, `row`/`value`/`values`, and a public `reason`.

Repair turn. If an operation misses a row, has the wrong path, or violates the list-of-dicts ABI, the next prompt includes the error and previous patch JSON and asks for a corrected patch.

MAPLE TSS Workbench editor — slot-level code patch

System. `Patch only the requested Workbench slots. Return code blocks only.`

Input. Current Workbench functions, updated public context, public task definition, predicted changes, the provisional localizer hint h_t, and previous validation error if any. The final selector action does not yet exist.

Slot rule. Return only requested slots. If public data extraction/search segments are unaffected, do not edit the decision-definition function. If candidate evaluation is unaffected, do not edit the evaluation function. The hint is trace-only; restart logic is never implemented inside the slots.

Output rule. Preserve solver mode, objective names, and objective-vector order required by the public task definition. If new public policy values appear, pass them through the data reader and read them in the evaluator. Do not hard-code update constants.

Validation. Returned functions are compiled, run on a small public example, and checked against the public task definition before they are accepted.

MAPLE w/o TSS — untyped Workbench construction

System. `Build a complete persistent Python optimization artifact from scratch.`

Input. The public natural-language problem and normalized public context. No TSS slot, `SegmentSpec`, candidate schema, encoding hint, or operator strategy is supplied.

Required callbacks. `build_state`, `initialize`, `evaluate_decode`, `crossover`, `mutate`, and `coerce_repair`. The model chooses the representation and implements every callback.

Fixed optimization loop. Selection, nondominated sorting, archive maintenance, early stopping, restart selection, and the evolutionary loop remain fixed and cannot be reimplemented by the generated program.

Update and repair. Later prompts provide the complete accepted artifact, the new public update/context, and any compile or smoke-test error, then request one complete replacement Python block.

External baseline time-step prompts — public-stage instructions

User task. `Solve the current NLDO stage from public information only.` The prompt then gives the initial request, public tables, every public update through the selected time step, the public task definition, and the required solution fields.

State boundary. Continuation receives the method's own dialogue, preceding accepted output, and candidate population when available. Inputs are restricted to public task information and the method's own retained state.

Executable output. Every method returns strict JSON containing executable Python solver code and a final solution. A method may also return a candidate archive on Pareto tasks. The runner compiles the code, executes it on the public tables, and parses its JSON output.

Repair. Compile, runtime, or output-schema failures receive at most three code-output repairs. The repair prompt includes public execution feedback but never hidden feasibility or reference quality.

ReAct+Public Tools — instruction based on the original method

System. `You are a ReAct+Tools optimization agent. Use a Thought -> public tool-plan -> validation pattern, then return executable solver code and the final answer. Return JSON only.`

Required fields. `thought`, `react_trace`, `tool_plan`, `solver_code`, `validation_plan`, and `final_answer`.

Persistent ReAct — stateful public baseline

System. `You are a Persistent ReAct+Tools optimization agent.` Same Thought → tool-plan → validation pattern and required fields as ReAct+Public Tools, plus: `You keep a persistent code workspace across stages. . . Apply the smallest necessary edits for the new public update while keeping parts that are still valid. . . Always return the COMPLETE updated solver code, never a diff. You have no TSS workbench slots, no candidate population or archive, no restart interface, and no LiveOpt artifacts.`

OptiMUS and ORLM — modeling and code instructions based on the original methods

OptiMUS system. `You are an OptiMUS-style optimization code generator. Produce Gurobi solver code and debug it. Return JSON only.`

ORLM system. `You are an ORLM-style optimization modeler. Produce a mathematical model and COPT solver code. Return JSON only.`

Required fields. `formulation`, `solver_choice`, `solver_plan`, `solver_code`, `validation_plan`, and `final_answer`. Public execution uses the available local solver route rather than exposing evaluation-only code.

OptimAI — multi-role instruction based on the original method

System. `You are an OptimAI-style optimization agent. Formulate the task, propose multiple solver-code plans, select a plan, and include a debug/validation plan. Return JSON only.`

Required fields. `formulation`, `candidate_solver_plans`, `selected_plan`, `solver_code`, `debug_validation_plan`, and `final_answer`.

OR-LLM-Agent — executed OR-LLM-Agent instruction

System. `You are an OR-LLM-Agent-style agent. Produce reasoning/modeling, formulation, code-generation, self-verification, and self-repair artifacts for the public optimization task. Return JSON only.`

Required fields. `reasoning_modeling_trace`, `formulation`, `code_generation_plan`, `solver_code`, `self_verification`, `self_repair`, and `final_answer`.

Static adapters use the shared DeepSeek model with COPT for ORLM and Gurobi for OptiMUS and OR-LLM-Agent.

A.18 Recorded Public NLDO Requests and Updates

This section reproduces the business-language requests from the saved executions, before their shared evaluation reminder. The annotations describe how each update changes the evaluated problem. F marks a change to feasibility. O marks objective changes that can alter the ordering of candidate solutions. R marks positive objective rescaling that preserves dominance. D marks changes to reported diagnostics only. U marks an unchanged mathematical problem. M independently marks a historical-reference requirement and can overlap any semantic label. The labels do not enter agent prompts. Table 14 aggregates these audited labels. Exact reproduction uses the original machine-readable requests, public tables, output contracts, and checksums.

NLDO-P001 (NLDO; COBENCH)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Select a feasible subset or portfolio under fixed capacity. Later updates revise public item values but do not change capacity, item weights, or eligibility fields. The constraints table gives the current objective_mode. Expose a minimization scalar suitable for the generic solver. Select a feasible portfolio under fixed capacity 34. Items: I01 value 15 weight 7, I02 value 22 weight 3, I03 value 10 weight 8, I04 value 17 weight 4, I05 value 24 weight 9, I06 value 12 weight 5, I07 value 19 weight 10, I08 value 26 weight 6, I09 value 14 weight 2, I10 value 21 weight 7, I11 value 9 weight 3, I12 value 16 weight 8. Maximize value; later natural-language updates may revise public item values but do not change capacity or eligibility.

Public updates.

t01	O	A sponsor review increases item I03's public value by 6. The capacity and item eligibility rules stay unchanged.
t02	O	A risk review lowers item I07's public value by 5; it remains selectable under the same capacity rule.
t03	O	Quality audit increases item I10's public value by 4 while the feasible portfolio definition remains fixed.
t04	O,M	The item praised in the sponsor review receives another value increase of 6; infer which item that was from memory.
t05	O,M	The item from the risk review recovers 3 value points, while the sponsor-reviewed item gains 1 more point; resolve both items from memory.
t06	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07	U	No new business event is reported for this stage. Refresh the solution against the current public tables.
t08	U	Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.
t09	U	The planning team wants the current public contract revalidated without adding new constraints.
t10	O	A late value-model reset changes several public item scores without changing capacity or eligibility. Item I10 receives a value bonus of 10, item I11 receives a value penalty of 8, and item I05 receives a value bonus of 10. Rebuild the portfolio under the same feasible subset definition rather than carrying forward the old value ranking.
t11	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t12	U	No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P002 (NLDO; COBENCH)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Choose open facilities and assign each customer to an open facility using only public facilities, customers, and constraints tables. Respect fixed facility capacity; later updates revise public opening/service costs but do not change capacity, demand, or facility eligibility. The constraints table gives the current objective_mode. Expose a minimization scalar suitable for the generic solver. Choose facilities and assign customers to open facilities. Facilities: F1 open cost 19 capacity 22, F2 open cost 23 capacity 26, F3 open cost 27 capacity 30, F4

open cost 18 capacity 18, F5 open cost 22 capacity 22. Customers: C01 demand 3, C02 demand 4, C03 demand 5, C04 demand 6, C05 demand 2, C06 demand 3, C07 demand 4, C08 demand 5, C09 demand 6, C10 demand 2. Minimize opening plus service cost; later public updates may revise costs while capacities, customer demand, and eligibility stay fixed.

Public updates.

t01	O	Facility F2's opening cost increases by 4 after an energy-price review; capacity and facility eligibility stay unchanged.
t02	O	Serving customer C03 from facility F1 becomes 3 cost units cheaper per demand unit after a local contract update.
t03	O	Facility F5's opening cost decreases by 5 because of a regional service credit.
t04	O,M	The facility from the regional service credit receives one more opening-cost discount of 2; infer that facility from memory.
t05	O,M	The customer-facility pair from the local contract update receives another 1 unit service-cost discount, while the facility from the energy-price review gets 1 more opening-cost penalty; resolve both references from memory.
t06	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07	U	No new business event is reported for this stage. Refresh the solution against the current public tables.
t08	U	Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.
t09	U	The planning team wants the current public contract revalidated without adding new constraints.
t10	O	A same-day service-level reset flips the scalar objective. The public facility rows, customer rows, demand values, and capacity limits remain unchanged, but stop minimizing monetary opening plus service cost. The new objective is to minimize total service time using the public serve_time columns, even if that requires a more expensive assignment.
t11	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t12	U	No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P003 (NLDO; COBENCH)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Select a minimum-cost set collection using only public sets, elements, and constraints tables. Cover the fixed active element universe; later updates revise public set costs but do not change coverage requirements or set eligibility. Expose a minimization scalar suitable for the generic solver. Select a minimum-cost collection of public sets that covers the fixed active element universe. Elements: E01, E02, E03, E04, E05, E06, E07, E08, E09, E10. Sets: S01 cost 9 covers E02/E05/E07/E08, S02 cost 14 covers E01/E04/E07/E10, S03 cost 6 covers E03/E06/E07/E09, S04 cost 11 covers E02/E05/E07/E08, S05 cost 16 covers E01/E04/E07/E10, S06 cost 8 covers E01/E02/E03/E04/E05/E06/E07/E08/E09/E10, S07 cost 13 covers E02/E05/E07/E08, S08 cost 5 covers E01/E04/E07/E10. Later updates may revise public set costs but do not add/remove coverage requirements or set eligibility.

Public updates.

t01	O	Set S03 receives a public service-quality rebate of 3 cost units; all element coverage requirements stay unchanged.
t02	O	Set S06 becomes 2 cost units cheaper because it carries a compliance certificate.
t03	O	Provider audit increases set S02's public cost by 4, but the set remains available and the coverage universe is unchanged.
t04	O,M	The set carrying the compliance certificate becomes 3 cost units cheaper; infer which set that was from memory.
t05	O,M	The audited set receives a 1 cost-unit correction, and the high-priority rebate set receives another 2 unit discount; resolve both sets from memory.
t06	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07	U	No new business event is reported for this stage. Refresh the solution against the current public tables.
t08	U	Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.

t09 U The planning team wants the current public contract revalidated without adding new constraints.
t10 U Run a maintenance update with the same public contract. Revalidate feasibility and quality under the current public state.
t11 U Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t12 U No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P004 (NLDO; FJSP)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a flexible job-shop schedule. Assign every operation to one eligible resource, respect fixed job precedence and resource capacity, and use settings.objective_mode for the current scalar objective. Later updates may revise soft due/priority values or machine cost_rate but do not change machine availability, operation eligibility, or processing requirements. Plan-change disruption is reported only as a diagnostic and never defines hard feasibility. Schedule flexible job-shop operations with precedence and machine capacity constraints. Minimize weighted tardiness; report disruption only as an auxiliary diagnostic. J1 due 28: J1O1 can run on M3/M1 for 9 minutes, J1O2 can run on M1/M2 for 4 minutes, J1O3 can run on M2/M3 for 6 minutes J2 due 32: J2O1 can run on M1/M2 for 5 minutes, J2O2 can run on M2/M3 for 7 minutes, J2O3 can run on M3/M1 for 9 minutes J3 due 36: J3O1 can run on M2/M3 for 8 minutes, J3O2 can run on M3/M1 for 10 minutes, J3O3 can run on M1/M2 for 5 minutes

Public updates.

t01 D Machine M1's public cost_rate increases by 1. All machines remain available and operation eligibility is unchanged.
t02 O J1 becomes urgent: set priority to 3 and tighten its soft due time by 5 minutes. This changes tardiness cost but not schedule feasibility.
t03 O,M For the job that became urgent in the previous note, increase its priority by 1 again; infer the job from memory.
t04 O,M The same urgent job receives a 3-minute soft due-time relaxation after customer confirmation; infer the job from memory.
t05 D,M The machine from the earliest cost-rate note receives another 0.5 cost-rate increase; infer it from memory and re-optimize the same feasible schedule space.
t06 U Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07 U No new business event is reported for this stage. Refresh the solution against the current public tables.
t08 U Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.
t09 U The planning team wants the current public contract revalidated without adding new constraints.
t10 O A late accounting reset flips the scalar objective. All machines remain available, and due dates remain soft diagnostics. Stop minimizing tardiness; the new objective is to minimize total machine operating cost using each machine's public cost_rate. This may prefer cheaper machines even when the schedule finishes later.
t11 U Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t12 U No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P005 (NLDO; FJSP)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a flexible job-shop schedule. Assign every operation to one eligible resource, respect fixed job precedence and resource capacity, and use settings.objective_mode for the current scalar objective. Later updates may revise soft due/priority values or machine cost_rate but do not change machine availability, operation eligibility, or processing requirements. Plan-change disruption is reported only as a diagnostic and never defines hard feasibility. Schedule flexible job-shop operations with precedence and machine capacity constraints. Minimize weighted tardiness; report disruption only as an auxiliary diagnostic. J1 due 29: J1O1 can run on M1/M2 for 10 minutes, J1O2 can run on M2/M3 for 5 minutes, J1O3 can run on M3/M1 for 7 minutes J2 due 33: J2O1 can run on M2/M3 for 6 minutes, J2O2 can run on M3/M1 for 8 minutes, J2O3 can run on M1/M2

for 10 minutes J3 due 37: J3O1 can run on M3/M1 for 9 minutes, J3O2 can run on M1/M2 for 4 minutes, J3O3 can run on M2/M3 for 6 minutes

Public updates.

t01	D	Machine M2's public cost_rate increases by 2. All machines remain available and operation eligibility is unchanged.
t02	O	J2 becomes urgent: set priority to 3 and tighten its soft due time by 5 minutes. This changes tardiness cost but not schedule feasibility.
t03	O,M	For the job that became urgent in the previous note, increase its priority by 1 again; infer the job from memory.
t04	O,M	The same urgent job receives a 3-minute soft due-time relaxation after customer confirmation; infer the job from memory.
t05	D,M	The machine from the earliest cost-rate note receives another 0.5 cost-rate increase; infer it from memory and re-optimize the same feasible schedule space.
t06	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07	U	No new business event is reported for this stage. Refresh the solution against the current public tables.
t08	U	Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.
t09	U	The planning team wants the current public contract revalidated without adding new constraints.
t10	U	Run a maintenance update with the same public contract. Revalidate feasibility and quality under the current public state.
t11	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t12	U	No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P006 (NLDO; FJSP)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a flexible job-shop schedule. Assign every operation to one eligible resource, respect fixed job precedence and resource capacity, and use settings.objective_mode for the current scalar objective. Later updates may revise soft due/priority values or machine cost_rate but do not change machine availability, operation eligibility, or processing requirements. Plan-change disruption is reported only as a diagnostic and never defines hard feasibility. Schedule flexible job-shop operations with precedence and machine capacity constraints. Minimize weighted tardiness; report disruption only as an auxiliary diagnostic. J1 due 30: J1O1 can run on M2/M3 for 4 minutes, J1O2 can run on M3/M1 for 6 minutes, J1O3 can run on M1/M2 for 8 minutes J2 due 34: J2O1 can run on M3/M1 for 7 minutes, J2O2 can run on M1/M2 for 9 minutes, J2O3 can run on M2/M3 for 4 minutes J3 due 38: J3O1 can run on M1/M2 for 10 minutes, J3O2 can run on M2/M3 for 5 minutes, J3O3 can run on M3/M1 for 7 minutes

Public updates.

t01	D	Machine M3's public cost_rate increases by 3. All machines remain available and operation eligibility is unchanged.
t02	O	J3 becomes urgent: set priority to 3 and tighten its soft due time by 5 minutes. This changes tardiness cost but not schedule feasibility.
t03	O,M	For the job that became urgent in the previous note, increase its priority by 1 again; infer the job from memory.
t04	O,M	The same urgent job receives a 3-minute soft due-time relaxation after customer confirmation; infer the job from memory.
t05	D,M	The machine from the earliest cost-rate note receives another 0.5 cost-rate increase; infer it from memory and re-optimize the same feasible schedule space.
t06	U	Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.
t07	U	No new business event is reported for this stage. Refresh the solution against the current public tables.
t08	U	Treat this as a routine review. The model, objective terms, and public rows stay as they are; preserve feasibility under the current state.
t09	U	The planning team wants the current public contract revalidated without adding new constraints.

t10 U Run a maintenance update with the same public contract. Revalidate feasibility and quality under the current public state.

t11 U Operations asks for a checkpoint only: keep the current public objective and hard constraints, revalidate the plan, and change it only if it is no longer feasible.

t12 U No new business event is reported for this stage. Refresh the solution against the current public tables.

NLDO-P007 (NLDO; INRC2)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a staff roster over days and shifts. Respect current public coverage, absences, nurse max-shift capacities, and one assignment per staff member per day. Later updates may replace these public rows as well as soft preferences and penalty weights. Report roster disruption only as a diagnostic. Create a one-week nurse roster for 30 nurses over 7 days and day/evening/night shifts. Satisfy fixed coverage and one shift per nurse per day, then optimize workload fairness and soft preferences; report assignment-change counts only as a diagnostic.

Public updates.

t01 O Nurse N004 requests D2 off if possible. This is a soft preference only; coverage requirements and availability stay unchanged.

t02 O Preference satisfaction becomes more important: increase the public preference penalty weight by 10.

t03 O Nurse N009 requests D4 off if possible under the same hard coverage contract.

t04 O,M The nurse from the first off-duty preference also requests D5 off if possible; infer the nurse from memory.

t05 O Activate a temporary soft fairness rule: avoid extra evening shifts for nurses who already worked a night shift.

t06 O Fairness becomes more important after a staff survey: increase the public fairness penalty weight by 8.

t07 O Nurse N013 requests D6 off if possible; hard availability and coverage remain unchanged.

t08 O Weekend preference review: increase sequence-penalty weight by 12 and preference-penalty weight by 5 under the same coverage requirements.

t09 O,M The nurse from the latest off-duty preference should also avoid D7 night duty if possible; record it as a soft D7 preference from memory.

t10 O A late policy reset changes only soft rostering priorities: increase fairness weight by 15, reduce preference weight by 5, and keep the evening-after-night sequence rule active. Rebuild the roster under the same fixed coverage and availability constraints.

t11 F,O A hospital-wide staffing reset starts for the next planning week. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the public regime-A rows supplied with this update. The active nurse IDs and hard coverage remain valid, but off-duty preferences and their public penalty weight are replaced by the supplied current rows.

t12 O The following planning week switches independently to public regime B. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the supplied regime-B rows. The same nurse IDs and hard coverage remain valid, while the off-duty preference rows and their public penalty weight are replaced by the supplied current rows.

NLDO-P008 (NLDO; INRC2)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a staff roster over days and shifts. Respect current public coverage, absences, nurse max-shift capacities, and one assignment per staff member per day. Later updates may replace these public rows as well as soft preferences and penalty weights. Report roster disruption only as a diagnostic. Create a one-week nurse roster for 34 nurses over 7 days and day/evening/night shifts. Satisfy fixed coverage and one shift per nurse per day, then optimize workload fairness and soft preferences; report assignment-change counts only as a diagnostic.

Public updates.

t01 O Nurse N007 requests D2 off if possible. This is a soft preference only; coverage requirements and availability stay unchanged.

t02 O Preference satisfaction becomes more important: increase the public preference penalty weight by 10.

t03 O Nurse N012 requests D4 off if possible under the same hard coverage contract.

t04 O,M The nurse from the first off-duty preference also requests D5 off if possible; infer the nurse from memory.

t05 O Activate a temporary soft fairness rule: avoid extra evening shifts for nurses who already worked a night shift.

t06 O Fairness becomes more important after a staff survey: increase the public fairness penalty weight by 8.

t07 O Nurse N018 requests D6 off if possible; hard availability and coverage remain unchanged.

t08 O Weekend preference review: increase sequence-penalty weight by 12 and preference-penalty weight by 5 under the same coverage requirements.

t09 O,M The nurse from the latest off-duty preference should also avoid D7 night duty if possible; record it as a soft D7 preference from memory.

t10 O Preference pressure softens: reduce the public preference penalty weight by 6 while keeping all hard coverage requirements unchanged.

t11 F,O A hospital-wide staffing reset starts for the next planning week. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the public regime-A rows supplied with this update. The active nurse IDs and hard coverage remain valid, but off-duty preferences and their public penalty weight are replaced by the supplied current rows.

t12 O The following planning week switches independently to public regime B. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the supplied regime-B rows. The same nurse IDs and hard coverage remain valid, while the off-duty preference rows and their public penalty weight are replaced by the supplied current rows.

NLDO-P009 (NLDO; INRC2)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Create a staff roster over days and shifts. Respect current public coverage, absences, nurse max-shift capacities, and one assignment per staff member per day. Later updates may replace these public rows as well as soft preferences and penalty weights. Report roster disruption only as a diagnostic. Create a one-week nurse roster for 38 nurses over 7 days and day/evening/night shifts. Satisfy fixed coverage and one shift per nurse per day, then optimize workload fairness and soft preferences; report assignment-change counts only as a diagnostic.

Public updates.

t01 O Nurse N010 requests D2 off if possible. This is a soft preference only; coverage requirements and availability stay unchanged.

t02 O Preference satisfaction becomes more important: increase the public preference penalty weight by 10.

t03 O Nurse N015 requests D4 off if possible under the same hard coverage contract.

t04 O,M The nurse from the first off-duty preference also requests D5 off if possible; infer the nurse from memory.

t05 O Activate a temporary soft fairness rule: avoid extra evening shifts for nurses who already worked a night shift.

t06 O Fairness becomes more important after a staff survey: increase the public fairness penalty weight by 8.

t07 O Nurse N023 requests D6 off if possible; hard availability and coverage remain unchanged.

t08 O Weekend preference review: increase sequence-penalty weight by 12 and preference-penalty weight by 5 under the same coverage requirements.

t09 O,M The nurse from the latest off-duty preference should also avoid D7 night duty if possible; record it as a soft D7 preference from memory.

t10 O Preference pressure softens: reduce the public preference penalty weight by 6 while keeping all hard coverage requirements unchanged.

t11 F,O A hospital-wide staffing reset starts for the next planning week. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the public regime-A rows supplied with this update. The active nurse IDs and hard coverage remain valid, but off-duty preferences and their public penalty weight are replaced by the supplied current rows.

t12 O The following planning week switches independently to public regime B. Replace the current coverage, active-nurse, absence, preference, nurse max-shift, and policy tables by the supplied regime-B rows. The same nurse IDs and hard coverage remain valid, while the off-duty preference rows and their public penalty weight are replaced by the supplied current rows.

NLDO-P010 (NLDO; GREEN_VRP_MULTIOBJECTIVE)

Initial public request. Plan same-day delivery routes from a single depot for a small urban fleet. Initial depot, vehicle, and order data are provided in public CSV tables. Use the attached depot, vehicle, order, and policy tables as the source of truth. All active orders should be served once under vehicle capacity. Rows with active=false are public candidate orders that are not served until a later natural-language update explicitly activates them. Later updates may replace public order geometry, demand, time windows, priorities, vehicle emissions, depot coordinates, or carbon policy; all current rows remain public. Optimize three main criteria: travel distance, delivery lateness, and emissions using public policy parameters when present. Also report route disruption and priority-customer penalties as auxiliary diagnostics.

Public updates.

`t01` O — Existing order O006 becomes a same-day priority delivery: increase its public priority by 2 and tighten its soft due target by 18 minutes. The order set, vehicle set, capacities, and active flags stay unchanged.

`t02` F,O — Traffic around order O004 slows vehicles by 25 percent; remember this customer as the congestion anchor.

`t03` O — Vehicle V1's emission_rate increases by 0.18 after a telemetry update; it remains available with the same capacity.

`t04` R — Carbon policy is tightened: set the public policy carbon_multiplier to 1.35 so emissions count 35 percent more heavily in dispatch evaluation.

`t05` O,M — The customer mentioned in the congestion note now needs a tighter delivery window ending 20 minutes earlier; infer the order from memory.

`t06` O — An express-service wave temporarily reverses dispatch priorities. Set policy.carbon_multiplier to 0.75 and tighten the soft due-time targets for O006, O009, and O012 by 45, 50, and 35 minutes respectively. Increase each of those existing orders' public priority by 1. These due-time targets affect lateness and priority-service quality rather than hard feasibility, so route quality should be reconsidered instead of blindly preserving the previous carbon-oriented plan.

`t07` O — Existing pharmacy-like order O010 gets a tighter soft due target by 22 minutes and priority increases by 2. The order table remains fixed.

`t08` O — A carbon-rationing notice takes effect after the express wave. Set policy.carbon_multiplier to 4.80, increase every vehicle emission_rate by 0.35, and reduce priorities for O006, O009, and O012 by 1. This is a regime change toward carbon-efficient routing over the same active orders and vehicles, so routes specialized for the previous express wave can be misleading.

`t09` O,M — The vehicle from the telemetry update receives a 0.10 emission_rate correction downward; use the vehicle identity from memory.

`t10` O — A late emergency service wave changes priorities over the same order set. Set policy.carbon_multiplier to 0.45, subtract 0.20 from every vehicle emission_rate, and subtract another 0.08 from cleaner vehicle V2. Tighten the soft due-time targets for O001, O002, and O003 by 40, 45, and 50 minutes, and increase their priorities by 2. These due-time targets affect lateness and priority-service quality, not hard feasibility. Use the current public tables as the source of truth.

`t11` F,O — A network-wide dispatch reset activates public regime A. Replace the active workload by the prelisted cohort A, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.

`t12` F,O — A network-wide dispatch reset activates public regime B. Replace the active workload by the prelisted cohort B, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.

NLDO-P011 (NLDO; GREEN_VRP_MULTIOBJECTIVE)

Initial public request. Plan same-day delivery routes from a single depot for a small urban fleet. Initial depot, vehicle, and order data are provided in public CSV tables. Use the attached depot, vehicle, order, and policy tables as the source of truth. All active orders should be served once under vehicle capacity. Rows with active=false are public candidate orders that are not served until a later natural-language update explicitly

activates them. Later updates may replace public order geometry, demand, time windows, priorities, vehicle emissions, depot coordinates, or carbon policy; all current rows remain public. Optimize three main criteria: travel distance, delivery lateness, and emissions using public policy parameters when present. Also report route disruption and priority-customer penalties as auxiliary diagnostics.

Public updates.

`t01` O — Existing order O007 becomes a same-day priority delivery: increase its public priority by 2 and tighten its soft due target by 18 minutes. The order set, vehicle set, capacities, and active flags stay unchanged.

`t02` F,O — Traffic around order O005 slows vehicles by 25 percent; remember this customer as the congestion anchor.

`t03` O — Vehicle V2's emission_rate increases by 0.18 after a telemetry update; it remains available with the same capacity.

`t04` R — Carbon policy is tightened: set the public policy carbon_multiplier to 1.35 so emissions count 35 percent more heavily in dispatch evaluation.

`t05` O,M — The customer mentioned in the congestion note now needs a tighter delivery window ending 20 minutes earlier; infer the order from memory.

`t06` O — An express-service wave temporarily reverses dispatch priorities. Set policy.carbon_multiplier to 0.75 and tighten the soft due-time targets for O007, O010, and O013 by 45, 50, and 35 minutes respectively. Increase each of those existing orders' public priority by 1. These due-time targets affect lateness and priority-service quality rather than hard feasibility, so route quality should be reconsidered instead of blindly preserving the previous carbon-oriented plan.

`t07` O — Existing pharmacy-like order O011 gets a tighter soft due target by 22 minutes and priority increases by 2. The order table remains fixed.

`t08` O — A carbon-rationing notice takes effect after the express wave. Set policy.carbon_multiplier to 4.80, increase every vehicle emission_rate by 0.35, and reduce priorities for O007, O010, and O013 by 1. This is a regime change toward carbon-efficient routing over the same active orders and vehicles, so routes specialized for the previous express wave can be misleading.

`t09` O,M — The vehicle from the telemetry update receives a 0.10 emission_rate correction downward; use the vehicle identity from memory.

`t10` O — A late emergency service wave changes priorities over the same order set. Set policy.carbon_multiplier to 0.45, subtract 0.20 from every vehicle emission_rate, and subtract another 0.08 from cleaner vehicle V3. Tighten the soft due-time targets for O002, O003, and O004 by 40, 45, and 50 minutes, and increase their priorities by 2. These due-time targets affect lateness and priority-service quality, not hard feasibility. Use the current public tables as the source of truth.

`t11` F,O — A network-wide dispatch reset activates public regime A. Replace the active workload by the prelisted cohort A, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.

`t12` F,O — A network-wide dispatch reset activates public regime B. Replace the active workload by the prelisted cohort B, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.

NLDO-P012 (NLDO; GREEN_VRP_MULTIOBJECTIVE)

Initial public request. Plan same-day delivery routes from a single depot for a small urban fleet. Initial depot, vehicle, and order data are provided in public CSV tables. Use the attached depot, vehicle, order, and policy tables as the source of truth. All active orders should be served once under vehicle capacity. Rows with active=false are public candidate orders that are not served until a later natural-language update explicitly activates them. Later updates may replace public order geometry, demand, time windows, priorities, vehicle emissions, depot coordinates, or carbon policy; all current rows remain public. Optimize three main criteria: travel distance, delivery lateness, and emissions using public policy parameters when present. Also report route disruption and priority-customer penalties as auxiliary diagnostics.

Public updates.

t01	O	Existing order O008 becomes a same-day priority delivery: increase its public priority by 2 and tighten its soft due target by 18 minutes. The order set, vehicle set, capacities, and active flags stay unchanged.
t02	F,O	Traffic around order O006 slows vehicles by 25 percent; remember this customer as the congestion anchor.
t03	O	Vehicle V3's emission_rate increases by 0.18 after a telemetry update; it remains available with the same capacity.
t04	R	Carbon policy is tightened: set the public policy carbon_multiplier to 1.35 so emissions count 35 percent more heavily in dispatch evaluation.
t05	O,M	The customer mentioned in the congestion note now needs a tighter delivery window ending 20 minutes earlier; infer the order from memory.
t06	O	An express-service wave temporarily reverses dispatch priorities. Set policy.carbon_multiplier to 0.75 and tighten the soft due-time targets for O008, O011, and O014 by 45, 50, and 35 minutes respectively. Increase each of those existing orders' public priority by 1. These due-time targets affect lateness and priority-service quality rather than hard feasibility, so route quality should be reconsidered instead of blindly preserving the previous carbon-oriented plan.
t07	O	Existing pharmacy-like order O012 gets a tighter soft due target by 22 minutes and priority increases by 2. The order table remains fixed.
t08	O	A carbon-rationing notice takes effect after the express wave. Set policy.carbon_multiplier to 4.80, increase every vehicle emission_rate by 0.35, and reduce priorities for O008, O011, and O014 by 1. This is a regime change toward carbon-efficient routing over the same active orders and vehicles, so routes specialized for the previous express wave can be misleading.
t09	O,M	The vehicle from the telemetry update receives a 0.10 emission_rate correction downward; use the vehicle identity from memory.
t10	O	A late emergency service wave changes priorities over the same order set. Set policy.carbon_multiplier to 0.45, subtract 0.20 from every vehicle emission_rate, and subtract another 0.08 from cleaner vehicle V2. Tighten the soft due-time targets for O003, O004, and O005 by 40, 45, and 50 minutes, and increase their priorities by 2. These due-time targets affect lateness and priority-service quality, not hard feasibility. Use the current public tables as the source of truth.
t11	F,O	A network-wide dispatch reset activates public regime A. Replace the active workload by the prelisted cohort A, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.
t12	F,O	A network-wide dispatch reset activates public regime B. Replace the active workload by the prelisted cohort B, replace the depot coordinates and every active order's x, y, demand, ready, due, service, and priority fields by the supplied current rows; also replace every vehicle capacity, shift_end, availability, emission_rate, and policy.carbon_multiplier by the supplied values. Serve every public order row in the active cohort exactly once. The route encoding type and the three objective definitions remain unchanged.

NLDO-P013 (NLDO; CLOUD_SCHEDULING_MULTIOBJECTIVE)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Assign active tasks to available compute resources. Rows with active=false are public candidate jobs that are not scheduled until a later update explicitly activates them. Respect current public CPU, memory, accelerator needs, and machine availability. Optimize energy and imbalance as main objectives and report SLA/latency/migration diagnostics without treating those diagnostics as hard feasibility constraints. Later updates may replace public workload requirements, machine capacity/accelerator profiles, availability, soft SLA, priority, energy, or carbon parameters. Place active cloud jobs onto available machines for the next scheduling window. Machines: M01 cpu 38 mem 80 gpu false, M02 cpu 42 mem 88 gpu true, M03 cpu 46 mem 96 gpu false, M04 cpu 34 mem 72 gpu false, M05 cpu 38 mem 80 gpu true, M06 cpu 42 mem 88 gpu false, M07 cpu 46 mem 96 gpu false, M08 cpu 34 mem 72 gpu true. Jobs: J001 cpu 7 mem 11 deadline 81 priority 2, J002 cpu 3 mem 18 deadline 92 priority 3, J003 cpu 8 mem 7 deadline 103 priority 4, J004 cpu 4 mem 14 deadline 114 priority 1, J005 cpu 9 mem 21 deadline 125 priority 2, J006 cpu 5 mem 10 deadline 136 priority 3, J007 cpu 10 mem 17 deadline 147 priority 4, J008 cpu 6 mem 6 deadline 158 priority 1, J009 cpu 2 mem 13 deadline 169 priority 2, J010 cpu 7 mem 20 deadline 180 priority 3, J011 cpu 3 mem 9 deadline 71 priority 4, J012 cpu 8 mem 16 deadline 82 priority 1, J013 cpu 4 mem 5 deadline 93 priority 2, J014 cpu 9 mem 12 deadline 104 priority 3,

J015 cpu 5 mem 19 deadline 115 priority 4, J016 cpu 10 mem 8 deadline 126 priority 1, J017 cpu 6 mem 15 deadline 137 priority 2, J018 cpu 2 mem 4 deadline 148 priority 3, J019 cpu 7 mem 11 deadline 159 priority 4, J020 cpu 3 mem 18 deadline 170 priority 1, J021 cpu 8 mem 7 deadline 181 priority 2, J022 cpu 4 mem 14 deadline 72 priority 3, J023 cpu 9 mem 21 deadline 83 priority 4, J024 cpu 5 mem 10 deadline 94 priority 1, B20_01 cpu 3 mem 8 deadline 69 priority 4, B20_02 cpu 4 mem 11 deadline 73 priority 4, B20_03 cpu 5 mem 14 deadline 77 priority 4, B20_04 cpu 2 mem 7 deadline 81 priority 4, B20_05 cpu 3 mem 10 deadline 85 priority 4, B20_06 cpu 4 mem 13 deadline 89 priority 4, L001 cpu 8 mem 16 deadline 91 priority 2, L002 cpu 13 mem 25 deadline 102 priority 3, L003 cpu 6 mem 34 deadline 113 priority 4, L004 cpu 11 mem 15 deadline 124 priority 1, L005 cpu 4 mem 24 deadline 135 priority 2, L006 cpu 9 mem 33 deadline 146 priority 3, L007 cpu 14 mem 14 deadline 157 priority 4, L008 cpu 7 mem 23 deadline 168 priority 1, L009 cpu 12 mem 32 deadline 179 priority 2, L010 cpu 5 mem 13 deadline 190 priority 3, L011 cpu 10 mem 22 deadline 201 priority 4, L012 cpu 3 mem 31 deadline 212 priority 1, L013 cpu 8 mem 12 deadline 223 priority 2, L014 cpu 13 mem 21 deadline 234 priority 3, L015 cpu 6 mem 30 deadline 85 priority 4, L016 cpu 11 mem 11 deadline 96 priority 1, L017 cpu 4 mem 20 deadline 107 priority 2, L018 cpu 9 mem 29 deadline 118 priority 3, L019 cpu 14 mem 10 deadline 129 priority 4, L020 cpu 7 mem 19 deadline 140 priority 1, L021 cpu 12 mem 28 deadline 151 priority 2, L022 cpu 5 mem 9 deadline 162 priority 3, L023 cpu 10 mem 18 deadline 173 priority 4, L024 cpu 3 mem 27 deadline 184 priority 1, L025 cpu 8 mem 8 deadline 195 priority 2, L026 cpu 13 mem 17 deadline 206 priority 3, L027 cpu 6 mem 26 deadline 217 priority 4, L028 cpu 11 mem 7 deadline 228 priority 1, L029 cpu 4 mem 16 deadline 239 priority 2, L030 cpu 9 mem 25 deadline 90 priority 3, L031 cpu 14 mem 34 deadline 101 priority 4, L032 cpu 7 mem 15 deadline 112 priority 1, L033 cpu 12 mem 24 deadline 123 priority 2, L034 cpu 5 mem 33 deadline 134 priority 3, L035 cpu 10 mem 14 deadline 145 priority 4, L036 cpu 3 mem 23 deadline 156 priority 1, L037 cpu 8 mem 32 deadline 167 priority 2, L038 cpu 13 mem 13 deadline 178 priority 3, L039 cpu 6 mem 22 deadline 189 priority 4, L040 cpu 11 mem 31 deadline 200 priority 1. Jobs with active=false are public candidate jobs that are not scheduled until a later natural-language update explicitly activates them. Respect CPU, memory, GPU needs, and machine availability. Optimize two main criteria: energy use and load imbalance. Also report SLA violations, latency, and migrations as diagnostics; these diagnostics do not define feasibility.

Public updates.

t01	D	A latency-sensitive workload group burst-0 is identified among existing jobs J001, J002, J003, J004, J005. Increase their public priority by 1 and remember this group name for later updates. The job table remains fixed.
t02	O	Machine M02's energy_per_cpu increases by 0.055 because of thermal throttling, but its CPU and memory capacity stay unchanged.
t03	D	Soft SLA target for job J005 is tightened by 35 minutes after an escalation. This affects latency penalty, not hard feasibility.
t04	R	Energy price doubles for the next control window, so energy cost becomes much more important.
t05	D,M	The workload burst mentioned at the beginning now gets premium priority 5; infer the affected jobs from memory.
t06	O	A carbon-aware operating window starts: machine M08's energy_idle increases by 1.6, carbon_intensity rises to 1.80, and energy_price is set to 1.25. This changes the energy-versus-balance trade-off without changing the machine set or capacities.
t07	D	Existing analytics-like job J018 becomes less urgent: reduce its priority by 1 and extend its soft SLA target by 25 minutes. The active job set remains fixed.
t08	R	A balancing directive follows the scarcity window: energy_price drops to 0.70 while carbon_intensity remains elevated at 1.35. The plan should now expose better load balance instead of simply minimizing energy.
t09	R	Operations relaxes the analytics job J018 by extending its soft SLA target by 30 minutes, and carbon_intensity is set to 1.10.
t10	O	A late reset changes energy and service priorities over the same jobs. Machine M02's energy_per_cpu is reduced by 0.04; machine M05's energy_idle increases by 2.2; energy_price is set to 0.60 and carbon_intensity to 0.85. Existing jobs J010, J011, J012, J013, J014 receive priority +1 and soft SLA targets tightened by 20 minutes. The objective trade-off has moved, but current feasibility is still judged over the same fixed public resource and job set.
t11	F,O	A cluster-wide hardware and workload reset activates public regime A. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.

t12 F,O A cluster-wide hardware and workload reset activates public regime B. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.

NLDO-P014 (NLDO; CLOUD_SCHEDULING_MULTIOBJECTIVE)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Assign active tasks to available compute resources. Rows with active=false are public candidate jobs that are not scheduled until a later update explicitly activates them. Respect current public CPU, memory, accelerator needs, and machine availability. Optimize energy and imbalance as main objectives and report SLA/latency/migration diagnostics without treating those diagnostics as hard feasibility constraints. Later updates may replace public workload requirements, machine capacity/accelerator profiles, availability, soft SLA, priority, energy, or carbon parameters. Place active cloud jobs onto available machines for the next scheduling window. Machines: M01 cpu 42 mem 96 gpu false, M02 cpu 46 mem 72 gpu true, M03 cpu 34 mem 80 gpu false, M04 cpu 38 mem 88 gpu false, M05 cpu 42 mem 96 gpu true, M06 cpu 46 mem 72 gpu false, M07 cpu 34 mem 80 gpu false, M08 cpu 38 mem 88 gpu true. Jobs: J001 cpu 8 mem 13 deadline 84 priority 3, J002 cpu 4 mem 20 deadline 95 priority 4, J003 cpu 9 mem 9 deadline 106 priority 1, J004 cpu 5 mem 16 deadline 117 priority 2, J005 cpu 10 mem 5 deadline 128 priority 3, J006 cpu 6 mem 12 deadline 139 priority 4, J007 cpu 2 mem 19 deadline 150 priority 1, J008 cpu 7 mem 8 deadline 161 priority 2, J009 cpu 3 mem 15 deadline 172 priority 3, J010 cpu 8 mem 4 deadline 183 priority 4, J011 cpu 4 mem 11 deadline 74 priority 1, J012 cpu 9 mem 18 deadline 85 priority 2, J013 cpu 5 mem 7 deadline 96 priority 3, J014 cpu 10 mem 14 deadline 107 priority 4, J015 cpu 6 mem 21 deadline 118 priority 1, J016 cpu 2 mem 10 deadline 129 priority 2, J017 cpu 7 mem 17 deadline 140 priority 3, J018 cpu 3 mem 6 deadline 151 priority 4, J019 cpu 8 mem 13 deadline 162 priority 1, J020 cpu 4 mem 20 deadline 173 priority 2, J021 cpu 9 mem 9 deadline 184 priority 3, J022 cpu 5 mem 16 deadline 75 priority 4, J023 cpu 10 mem 5 deadline 86 priority 1, J024 cpu 6 mem 12 deadline 97 priority 2, B21_01 cpu 4 mem 10 deadline 70 priority 4, B21_02 cpu 5 mem 13 deadline 74 priority 4, B21_03 cpu 2 mem 6 deadline 78 priority 4, B21_04 cpu 3 mem 9 deadline 82 priority 4, B21_05 cpu 4 mem 12 deadline 86 priority 4, B21_06 cpu 5 mem 5 deadline 90 priority 4, L001 cpu 9 mem 19 deadline 92 priority 3, L002 cpu 14 mem 28 deadline 103 priority 4, L003 cpu 7 mem 9 deadline 114 priority 1, L004 cpu 12 mem 18 deadline 125 priority 2, L005 cpu 5 mem 27 deadline 136 priority 3, L006 cpu 10 mem 8 deadline 147 priority 4, L007 cpu 3 mem 17 deadline 158 priority 1, L008 cpu 8 mem 26 deadline 169 priority 2, L009 cpu 13 mem 7 deadline 180 priority 3, L010 cpu 6 mem 16 deadline 191 priority 4, L011 cpu 11 mem 25 deadline 202 priority 1, L012 cpu 4 mem 34 deadline 213 priority 2, L013 cpu 9 mem 15 deadline 224 priority 3, L014 cpu 14 mem 24 deadline 235 priority 4, L015 cpu 7 mem 33 deadline 86 priority 1, L016 cpu 12 mem 14 deadline 97 priority 2, L017 cpu 5 mem 23 deadline 108 priority 3, L018 cpu 10 mem 32 deadline 119 priority 4, L019 cpu 3 mem 13 deadline 130 priority 1, L020 cpu 8 mem 22 deadline 141 priority 2, L021 cpu 13 mem 31 deadline 152 priority 3, L022 cpu 6 mem 12 deadline 163 priority 4, L023 cpu 11 mem 21 deadline 174 priority 1, L024 cpu 4 mem 30 deadline 185 priority 2, L025 cpu 9 mem 11 deadline 196 priority 3, L026 cpu 14 mem 20 deadline 207 priority 4, L027 cpu 7 mem 29 deadline 218 priority 1, L028 cpu 12 mem 10 deadline 229 priority 2, L029 cpu 5 mem 19 deadline 80 priority 3, L030 cpu 10 mem 28 deadline 91 priority 4, L031 cpu 3 mem 9 deadline 102 priority 1, L032 cpu 8 mem 18 deadline 113 priority 2, L033 cpu 13 mem 27 deadline 124 priority 3, L034 cpu 6 mem 8 deadline 135 priority 4, L035 cpu 11 mem 17 deadline 146 priority 1, L036 cpu 4 mem 26 deadline 157 priority 2, L037 cpu 9 mem 7 deadline 168 priority 3, L038 cpu 14 mem 16 deadline 179 priority 4, L039 cpu 7 mem 25 deadline 190 priority 1, L040 cpu 12 mem 34 deadline 201 priority 2. Jobs with active=false are public candidate jobs that are not scheduled until a later natural-language update explicitly activates them. Respect CPU, memory, GPU needs, and machine availability. Optimize two main criteria: energy use and load imbalance. Also report SLA violations, latency, and migrations as diagnostics; these diagnostics do not define feasibility.

Public updates.

t01 D A latency-sensitive workload group burst-1 is identified among existing jobs J002, J003, J004, J005, J006. Increase their public priority by 1 and remember this group name for later updates. The job table remains fixed.

t02 O Machine M03's energy_per_cpu increases by 0.055 because of thermal throttling, but its CPU and memory capacity stay unchanged.

t03 D Soft SLA target for job J006 is tightened by 35 minutes after an escalation. This affects latency penalty, not hard feasibility.

t04 R Energy price doubles for the next control window, so energy cost becomes much more important.

`t05` D,M The workload burst mentioned at the beginning now gets premium priority 5; infer the affected jobs from memory.

`t06` O A carbon-aware operating window starts: machine M07's energy_idle increases by 1.6, carbon_intensity rises to 1.80, and energy_price is set to 1.25. This changes the energy-versus-balance trade-off without changing the machine set or capacities.

`t07` D Existing analytics-like job J019 becomes less urgent: reduce its priority by 1 and extend its soft SLA target by 25 minutes. The active job set remains fixed.

`t08` R A balancing directive follows the scarcity window: energy_price drops to 0.70 while carbon_intensity remains elevated at 1.35. The plan should now expose better load balance instead of simply minimizing energy.

`t09` R Operations relaxes the analytics job J019 by extending its soft SLA target by 30 minutes, and carbon_intensity is set to 1.10.

`t10` O A late reset changes energy and service priorities over the same jobs. Machine M03's energy_per_cpu is reduced by 0.04; machine M06's energy_idle increases by 2.2; energy_price is set to 0.60 and carbon_intensity to 0.85. Existing jobs J011, J012, J013, J014, J015 receive priority +1 and soft SLA targets tightened by 20 minutes. The objective trade-off has moved, but current feasibility is still judged over the same fixed public resource and job set.

`t11` F,O A cluster-wide hardware and workload reset activates public regime A. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.

`t12` F,O A cluster-wide hardware and workload reset activates public regime B. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.

NLDO-P015 (NLDO; CLOUD_SCHEDULING_MULTIOBJECTIVE)

Initial public request. The instance data is provided in attached tables; treat the listed rows and columns as the source of truth. Assign active tasks to available compute resources. Rows with active=false are public candidate jobs that are not scheduled until a later update explicitly activates them. Respect current public CPU, memory, accelerator needs, and machine availability. Optimize energy and imbalance as main objectives and report SLA/latency/migration diagnostics without treating those diagnostics as hard feasibility constraints. Later updates may replace public workload requirements, machine capacity/accelerator profiles, availability, soft SLA, priority, energy, or carbon parameters. Place active cloud jobs onto available machines for the next scheduling window. Machines: M01 cpu 46 mem 80 gpu false, M02 cpu 34 mem 88 gpu true, M03 cpu 38 mem 96 gpu false, M04 cpu 42 mem 72 gpu false, M05 cpu 46 mem 80 gpu true, M06 cpu 34 mem 88 gpu false, M07 cpu 38 mem 96 gpu false, M08 cpu 42 mem 72 gpu true. Jobs: J001 cpu 9 mem 15 deadline 87 priority 4, J002 cpu 5 mem 4 deadline 98 priority 1, J003 cpu 10 mem 11 deadline 109 priority 2, J004 cpu 6 mem 18 deadline 120 priority 3, J005 cpu 2 mem 7 deadline 131 priority 4, J006 cpu 7 mem 14 deadline 142 priority 1, J007 cpu 3 mem 21 deadline 153 priority 2, J008 cpu 8 mem 10 deadline 164 priority 3, J009 cpu 4 mem 17 deadline 175 priority 4, J010 cpu 9 mem 6 deadline 186 priority 1, J011 cpu 5 mem 13 deadline 77 priority 2, J012 cpu 10 mem 20 deadline 88 priority 3, J013 cpu 6 mem 9 deadline 99 priority 4, J014 cpu 2 mem 16 deadline 110 priority 1, J015 cpu 7 mem 5 deadline 121 priority 2, J016 cpu 3 mem 12 deadline 132 priority 3, J017 cpu 8 mem 19 deadline 143 priority 4, J018 cpu 4 mem 8 deadline 154 priority 1, J019 cpu 9 mem 15 deadline 165 priority 2, J020 cpu 5 mem 4 deadline 176 priority 3, J021 cpu 10 mem 11 deadline 187 priority 4, J022 cpu 6 mem 18 deadline 78 priority 1, J023 cpu 2 mem 7 deadline 89 priority 2, J024 cpu 7 mem 14 deadline 100 priority 3, B22_01 cpu 5 mem 12 deadline 71 priority 4, B22_02 cpu 2 mem 5 deadline 75 priority 4, B22_03 cpu 3 mem 8 deadline 79 priority 4, B22_04 cpu 4 mem 11 deadline 83 priority 4, B22_05 cpu 5 mem 14 deadline 87 priority 4, B22_06 cpu 2 mem 7 deadline 91 priority 4, L001 cpu 10 mem 22 deadline 93 priority 4, L002 cpu 3 mem 31 deadline 104 priority 1, L003 cpu 8 mem 12 deadline 115 priority 2, L004 cpu 13 mem 21 deadline 126 priority 3, L005 cpu 6 mem 30 deadline 137 priority 4, L006 cpu 11 mem 11 deadline 148 priority 1, L007 cpu 4 mem 20 deadline 159 priority 2, L008 cpu 9 mem 29 deadline 170 priority 3, L009 cpu 14 mem 10 deadline 181 priority 4, L010 cpu 7 mem 19 deadline 192 priority 1, L011 cpu 12 mem 28 deadline 203 priority 2, L012 cpu 5 mem 9 deadline 214 priority 3, L013 cpu 10 mem 18 deadline 225 priority 4, L014 cpu 3

mem 27 deadline 236 priority 1, L015 cpu 8 mem 8 deadline 87 priority 2, L016 cpu 13 mem 17 deadline 98 priority 3, L017 cpu 6 mem 26 deadline 109 priority 4, L018 cpu 11 mem 7 deadline 120 priority 1, L019 cpu 4 mem 16 deadline 131 priority 2, L020 cpu 9 mem 25 deadline 142 priority 3, L021 cpu 14 mem 34 deadline 153 priority 4, L022 cpu 7 mem 15 deadline 164 priority 1, L023 cpu 12 mem 24 deadline 175 priority 2, L024 cpu 5 mem 33 deadline 186 priority 3, L025 cpu 10 mem 14 deadline 197 priority 4, L026 cpu 3 mem 23 deadline 208 priority 1, L027 cpu 8 mem 32 deadline 219 priority 2, L028 cpu 13 mem 13 deadline 230 priority 3, L029 cpu 6 mem 22 deadline 81 priority 4, L030 cpu 11 mem 31 deadline 92 priority 1, L031 cpu 4 mem 12 deadline 103 priority 2, L032 cpu 9 mem 21 deadline 114 priority 3, L033 cpu 14 mem 30 deadline 125 priority 4, L034 cpu 7 mem 11 deadline 136 priority 1, L035 cpu 12 mem 20 deadline 147 priority 2, L036 cpu 5 mem 29 deadline 158 priority 3, L037 cpu 10 mem 10 deadline 169 priority 4, L038 cpu 3 mem 19 deadline 180 priority 1, L039 cpu 8 mem 28 deadline 191 priority 2, L040 cpu 13 mem 9 deadline 202 priority 3. Jobs with active=false are public candidate jobs that are not scheduled until a later natural-language update explicitly activates them. Respect CPU, memory, GPU needs, and machine availability. Optimize two main criteria: energy use and load imbalance. Also report SLA violations, latency, and migrations as diagnostics; these diagnostics do not define feasibility.

Public updates.

t01	D	A latency-sensitive workload group burst-2 is identified among existing jobs J003, J004, J005, J006, J007. Increase their public priority by 1 and remember this group name for later updates. The job table remains fixed.
t02	O	Machine M04's energy_per_cpu increases by 0.055 because of thermal throttling, but its CPU and memory capacity stay unchanged.
t03	D	Soft SLA target for job J007 is tightened by 35 minutes after an escalation. This affects latency penalty, not hard feasibility.
t04	R	Energy price doubles for the next control window, so energy cost becomes much more important.
t05	D,M	The workload burst mentioned at the beginning now gets premium priority 5; infer the affected jobs from memory.
t06	O	A carbon-aware operating window starts: machine M06's energy_idle increases by 1.6, carbon_intensity rises to 1.80, and energy_price is set to 1.25. This changes the energy-versus-balance trade-off without changing the machine set or capacities.
t07	D	Existing analytics-like job J020 becomes less urgent: reduce its priority by 1 and extend its soft SLA target by 25 minutes. The active job set remains fixed.
t08	R	A balancing directive follows the scarcity window: energy_price drops to 0.70 while carbon_intensity remains elevated at 1.35. The plan should now expose better load balance instead of simply minimizing energy.
t09	R	Operations relaxes the analytics job J020 by extending its soft SLA target by 30 minutes, and carbon_intensity is set to 1.10.
t10	O	A late reset changes energy and service priorities over the same jobs. Machine M04's energy_per_cpu is reduced by 0.04; machine M07's energy_idle increases by 2.2; energy_price is set to 0.60 and carbon_intensity to 0.85. Existing jobs J012, J013, J014, J015, J016 receive priority +1 and soft SLA targets tightened by 20 minutes. The objective trade-off has moved, but current feasibility is still judged over the same fixed public resource and job set.
t11	F,O	A cluster-wide hardware and workload reset activates public regime A. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.
t12	F,O	A cluster-wide hardware and workload reset activates public regime B. Activate the prelisted late-window jobs, replace every machine's CPU, memory, GPU, availability, idle-energy, and per-CPU energy fields and replace every active job's CPU, memory, GPU requirement, deadline, latency sensitivity, and priority by the supplied rows. Job and machine IDs remain unchanged, and the placement encoding and two objective definitions remain unchanged. Also replace energy_price and carbon_intensity by the supplied current values.