

Where should control sit? Reliability-cost trade-offs in delegating water distribution network optimisation to LLM agents

Jian Wang^a, Shuming Liu^b, Guangtao Fu^{a,*}, Dragan Savic^{a,c,*}

^a Centre for Water Systems, University of Exeter, Exeter, EX4 4QF, United Kingdom

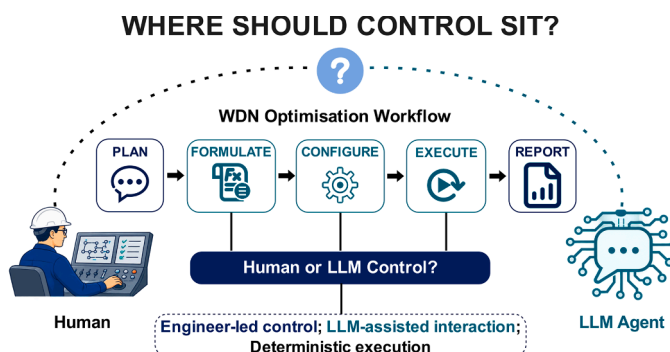
^b School of Environment, Tsinghua University, Beijing, 100084, China

^c KWR Water Research Institute, Nieuwegein, 3430 BB, the Netherlands

HIGHLIGHTS

- Control allocation governs LLM automation reliability more than solution quality.
- A configurable workflow-agent architecture tests WDN optimisation on C-Town.
- Delegating formulation raises token use sixfold without improving outcomes.
- Failures concentrate at delegated nodes and worsen as model capability falls.
- Workflow-dominant control remains viable on a 14B-parameter local model.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:

Automation
Water distribution network
Optimisation
Large language models
Agentic workflows
Decision support

ABSTRACT

Computational optimisation is now routinely used to support the design and operation of water distribution networks (WDNs). However, formulating optimisation problems, configuring solution workflows and interpreting results still depend heavily on manual effort, making even standard studies time-consuming to reproduce and difficult to scale. Large language models (LLMs) offer new opportunities to automate these tasks, but their effective use depends on a central design question: which steps should be delegated to an autonomous LLM agent, and which should remain deterministic workflow steps. This study quantifies the reliability and resource cost of delegating individual stages of a WDN optimisation pipeline to an LLM agent. We reformulate an end-to-end optimisation pipeline as a sequence of decision nodes, each of which can operate either as a deterministic workflow step or as a ReAct (Reasoning + Acting) agent, so that the locus of control is varied while the underlying optimisation task remains unchanged. Two representative tasks, i.e., single-objective pump energy minimisation and multi-objective energy-resilience optimisation problems, are evaluated on the benchmark C-Town network across five LLMs spanning an order of magnitude in model scale, with control transferred one node at a time over 250 runs. The results obtained show that control allocation affects reliability and computational cost far more than optimisation quality. Solution quality was largely insensitive to the control model, with variations of about 1% in pump energy and 13% in hypervolume, and neither the deterministic workflow nor the ReAct-agent configuration consistently outperformed the other. In contrast, delegating individual nodes

* Corresponding authors at: Centre for Water Systems, University of Exeter, Exeter, EX4 4QF, United Kingdom.

E-mail addresses: g.fu@exeter.ac.uk (G. Fu), d.savic@exeter.ac.uk (D. Savic).

<https://doi.org/10.1016/j.watres.2026.126823>

Received 29 June 2026; Received in revised form 15 August 2026; Accepted 28 August 2026

Available online 28 August 2026

0043-1354/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

to the ReAct-agent substantially increased resource use: at the formulation node, token consumption rose by roughly sixfold and the number of model calls by an order of magnitude. Workflow failures were structural rather than random and were concentrated at the delegated nodes. Full autonomy was reliable only for the largest LLMs, whereas workflow-dominant configurations remained viable down to a 14B-parameters LLM. These findings indicate that, for well-specified engineering optimisation tasks, LLMs autonomy should be extended selectively rather than by default. More broadly, by establishing where autonomy can be trusted and where it cannot, this work lays a foundation for the reliable, reproducible and auditable LLM-driven decision support on which trustworthy automation of water systems will depend.

1. Introduction

Water distribution networks (WDNs) are critical infrastructure that underpin public health, economic activity, and urban resilience (Brahmbhatt et al., 2023; Cassiolato et al., 2024; Dini et al., 2022). Their operation has become more challenging as systems are expected to deliver higher levels of performance under increasingly constrained conditions (Ortega-Ballesteros et al., 2021; Taiwo et al., 2025). In this context, optimisation has become an important tool for operational decision-making (Sarbu, 2021; Sarbu and Popa-Albu, 2023) and longer-term planning in WDNs, supporting tasks such as pump scheduling, pressure management, network design and rehabilitation, and the balancing of competing performance objectives (Awwalu et al., 2024; Sarbu, 2021).

In practice, despite their diversity, most optimisation tasks have a strikingly similar procedure: a problem is formulated from an engineering question by engineers, an optimisation model is used to derive candidate solutions, these solutions are evaluated against hydraulic behaviour using a simulation model, and the results are interpreted to inform a decision. In current practice this procedure is largely carried out manually, encoded in bespoke scripts and dependent on substantial expert judgement, so that even routine analyses are costly to set up and hard to reproduce or transfer when the same logic must be re-applied across networks, objectives or scenarios (Dobson et al., 2025; Horsburgh et al., 2025; Zanutto et al., 2025). Reducing this manual burden, without sacrificing the transparency and repeatability that engineering analysis demands, is therefore a long-standing obstacle to scaling model-based decision support in the water sector.

Large language models (LLMs) have recently emerged as a way to lower this barrier, because they can translate natural language requests into structured actions, reason over intermediate results, and interact with users in plain language (Fu, 2025; Sela et al., 2025; Taiwo et al., 2025; Xu et al., 2025). Early water-sector studies have begun to use them as natural language interfaces to hydraulic models, as assistants for information extraction, and as components of increasingly autonomous multi-agent systems (Goldshtein et al., 2025; Hosseini et al., 2026; Wang et al., 2026). Two contrasting approaches to embedding LLMs have crystallised in the wider literature. In a workflow, the engineer defines the procedure in advance: the LLM is invoked at fixed points to handle specific sub-tasks such as interpreting a user request or summarising a result, while the overall sequence remains under human control (Anthropic, 2024; Li et al., 2024). In an agent, decision authority is transferred to the LLM itself: given a goal and a set of available actions, the agent decides what to do next, which actions to take, and when the task is complete, without following any prescribed order (Yao et al., 2023; Yu et al., 2025). Across much of the recent literature, the prevailing direction has been towards progressively greater autonomy.

For engineering optimisation, however, greater autonomy is not self-evidently better. The same freedom that lets an agent adapt to an open-ended request also lets small differences in its reasoning propagate into different tool calls, different configurations and, ultimately, different and harder-to-audit outcomes (Liu et al., 2025; Zhao et al., 2025). The pertinent design question is therefore not only whether to use an LLM, but how much procedural control should remain in a fixed workflow and how much should be delegated to the agent. Yet, for WDN optimisation,

the reliability, cost and reproducibility consequences of this stage-level delegation, and their dependence on model capability, remain insufficiently quantified.

This study quantifies the consequences of executing individual stages of a WDN optimisation pipeline as deterministic workflow steps versus autonomous LLM agents. To do so, the choice of executing each stage as a deterministic workflow step or as an autonomous agent is treated as a controlled experimental variable rather than as a fixed design choice. An end-to-end WDN optimisation pipeline is recast as a set of nodes, comprising planning, parameter elicitation, experiment design, execution and reporting. The three execution-critical nodes can each operate either as deterministic workflow steps or as ReAct agents equipped with matched tools, so that the locus of control changes while the underlying capability remains constant. Using the C-Town benchmark and two representative operational tasks, namely single-objective pump-energy minimisation and multi-objective energy-resilience optimisation, control is transferred one node at a time across five LLMs spanning roughly an order of magnitude in scale. The consequences are evaluated in terms of solution quality, token and model-call cost, and the rate and structure of failure. The results characterise how delegating each stage affects solution quality, resource cost and the rate and structure of failure, and how these effects depend on model capability, for well-specified operational optimisation tasks.

2. Methodology

Automating WDN optimisation with an LLM requires deciding how much of the procedure should be prescribed as a deterministic workflow and how much delegated to the LLM agent. This section develops and characterises a configurable optimisation pipeline that makes this allocation an explicit design decision. LLMs are embedded as language-based components that interpret the human's intent (e.g., minimise pump energy over a 24-hour schedule), elicit any unspecified optimisation parameters (e.g., which pumps to control or the target service pressure), and reason over intermediate outputs, while hydraulic simulation and optimisation remain deterministic and predefined, i.e., executed through models (Acharya et al., 2025; Guo et al., 2024). The remainder of this section is organised as follows: Section 2.1 defines the two representative WDN optimisation tasks used for evaluation; Section 2.2 describes the configurable workflow-agent architecture; Section 2.3 explains the shared-state and prompt construction mechanism; Section 2.4 defines the metrics used to evaluate performance; and Section 2.5 details the algorithm catalogue, LLM selection, and implementation settings.

2.1. Problem definition for representative WDN optimisation tasks

This study considers two representative operational optimisation objectives in WDNs: minimising pump energy consumption and maximising system resilience, measured by the modified resilience index (MRI) proposed by Jayaram and Srinivasan (2008). These two objectives expose a direct trade-off between energy efficiency and hydraulic performance, span single- and multi-objective settings, and together provide a compact, reproducible basis for evaluating how different allocations of control between the human-defined procedure and the

LLM affect automation quality, reliability and computational cost.

Pump energy minimisation: this task addresses the reduction of pump energy use during network operation over a fixed simulation horizon (Kluse et al., 2020), formulated as:

$$\text{Minimise } J_{\text{energy}} = \sum_{t=1}^T \sum_{p \in P} \frac{\rho g Q_{p,t} \Delta H_{p,t}}{\eta_{p,t}} \Delta t \quad (1)$$

where ρ is the water density, g is the gravitational acceleration, $Q_{p,t}$ is the flow rate through pump p at time t , $\Delta H_{p,t}$ is the corresponding pump head gain, $\eta_{p,t}$ is the pump efficiency, and Δt is the simulation time step.

System resilience maximisation: The MRI quantifies the ratio of surplus hydraulic power available at demand nodes above the required service head to the minimum hydraulic power needed to satisfy demand (Jayaram and Srinivasan, 2008). It is averaged over the simulation horizon:

$$\text{Maximise } \frac{1}{k} \sum_{t=1}^k \text{MRI}_t \quad (2)$$

$$\text{MRI}_t = \frac{\sum_{n=1}^N Q_n^{\text{req}} (H_n^t - H_n^{\text{req}})}{\sum_{n=1}^N Q_n^{\text{req}} H_n^{\text{req}}} \quad (3)$$

where k is the number of simulation time steps, N the number of demand

nodes, Q_n^{req} the required demand at node n , H_n^t the simulated nodal head at time t , and H_n^{req} the required service head. MRI increases monotonically with surplus nodal head, making it well suited to pump scheduling problems in which nodal pressures vary dynamically across the operating cycle.

In both tasks (single and multi-objective optimisation), the decision variables are the speed multipliers of the variable-speed pumps, applied as time-varying settings over the operating cycle. A hard hydraulic constraint enforces non-negative pressure at every node throughout the simulation horizon, and the required service pressure is set to 20 m for all demand nodes (Giustolisi et al., 2014).

2.2. A configurable workflow-agent architecture for WDN optimisation

Automating a WDN optimisation pipeline with an LLM requires a fundamental design decision: how much of the procedure should be prescribed by the human, and how much should be left for the LLM to determine at runtime. Two contrasting approaches are possible (Fig. 1). In the workflow mode (WM), the human defines every step in advance; the LLM is invoked at fixed points within a predefined sequence, each call operates within a constrained and bounded scope, and control passes between steps deterministically, so the same inputs always produce the same execution path (Anthropic, 2024; Li et al., 2024). In the ReAct-agent mode (RM), this authority is transferred to the LLM itself:

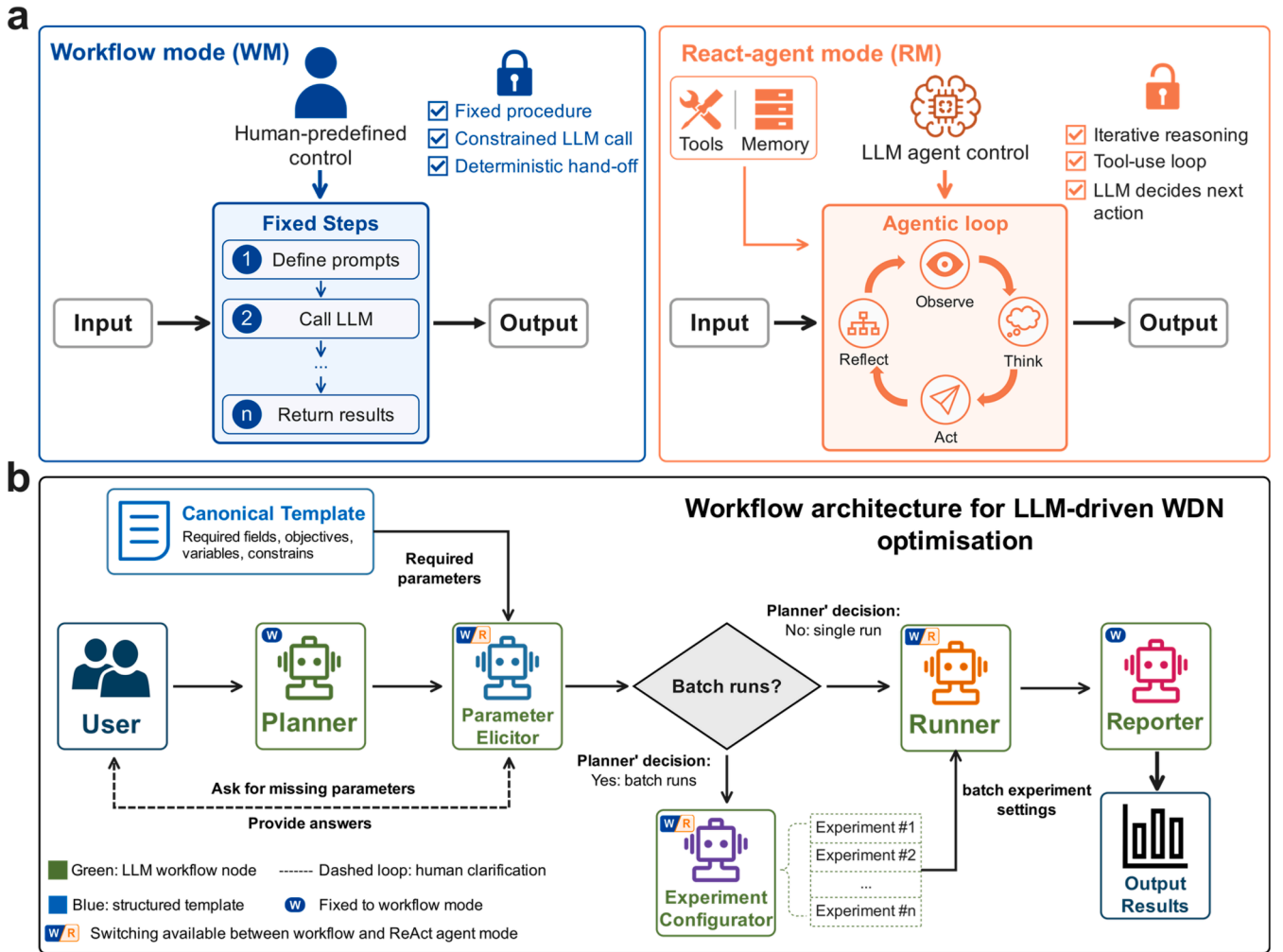


Fig. 1. Configurable workflow-agent architecture for LLM-driven WDN optimisation. (a) Workflow mode retains control in fixed, human-prescribed steps, whereas ReAct-agent mode delegates action selection to an iterative LLM tool-use loop. (b) The five-node optimisation pipeline, in which parameter elicitation, experiment configuration and running can each switch between workflow and agent modes, while planning and reporting remain fixed workflow steps.

given a goal and a set of tools, the agent iterates through an observe-think-act loop, deciding at each step which tool to call and when the task is complete, without any prescribed order (Yu et al., 2025). The key distinction is therefore not what the LLM can do, but where control over the procedure resides. Workflow mode preserves execution transparency and reproducibility by constraining the LLM to predefined, auditable steps, whereas RM grants the model authority over its own procedure at the cost of less predictable and harder-to-audit execution (Qiu et al., 2025; Shin et al., 2025).

To implement this control allocation as an explicit experimental variable, the WDN optimisation pipeline is constructed as a five-node graph built around a canonical configuration template, as shown in Fig. 1. The pipeline proceeds as follows.

Planner (WM): the user's natural-language query is interpreted and converted into a high-level optimisation plan, clarifying the task type, the objective (or objectives) involved, and the overall problem formulation to be specified in subsequent stages. This node always operates in workflow mode.

Parameter Elicitor (WM/RM): guided by the plan and the canonical configuration template (Table S1), this node iteratively queries the user to elicit and validate each required input, including objectives, decision variables, constraints, algorithm selection and stopping criteria, proceeding only when the specification is complete and passes validation against the shared configuration schema (Section 2.5).

Experiment Configurator invoked when the Planner identifies an experimental design requirement (WM/RM): the completed specification is taken as a baseline and expanded into a set of optimisation runs, each configured according to the user's experimental requirements.

Runner (WM/RM): the configuration assembled by the Parameter Elicitor and Experiment Configurator is executed by running the specified population-based metaheuristic search with the EPANET-based hydraulic solver (Rossman et al., 2020), which evaluates each candidate solution hydraulically. The Runner produces result files and visual outputs that are then passed to the Reporter; for multi-objective tasks, it returns the Pareto-optimal solution set.

Reporter (WM): execution outputs and the optimisation plan are synthesised into a structured, interpretable report that summarises numerical results and, where available, visualisations for decision support. This node always operates in workflow mode.

The Planner and Reporter, which sit at the user-computation interface, are held in WM mode throughout, as each performs a single, bounded input-to-output transformation that requires no iterative decision-making or tool use. The three execution-critical nodes (Parameter Elicitor, Experiment Configurator and Runner), by contrast, involve multi-step reasoning or interaction and can each be engaged independently between WM and RM modes, making control allocation an explicit experimental variable.

To ensure a fair comparison, each node operating in RM mode is given a bounded tool set that exactly reproduces the capabilities of its workflow counterpart, as shown in Table 1. The parameter node may ask the user a clarifying question, write a specification field, validate the specification against the shared schema and finalise it; the experiment node may read the baseline, propose, list or remove experiments and commit the final set; and the running node may execute a named experiment and write results and figure metadata to the shared state. The underlying capabilities of each node are therefore identical across modes; the sole difference is whether these operations are sequenced by the predefined workflow or selected autonomously by the agent.

2.3. Shared-state prompt construction and context propagation

Both control modes are coordinated through a single shared workflow state that acts as the sole source of truth across the optimisation pipeline, as illustrated in Fig. 2. This state stores the user query, the planner-generated plan and task metadata, the elicited optimisation

Table 1

Tools registered for the three configurable nodes when operated in ReAct-agent mode. Each tool set reproduces the capabilities of the corresponding workflow node, so that only the locus of control differs between modes.

Configurable node (position)	ReAct tool	Function
Parameter elicitation (1, P)	ask_user	Pause the optimisation pipeline and ask the user one clarifying question, returning the answer.
	set_parameter	Write a single field of the optimisation specification (objective, variable, algorithm, etc.).
	validate_configuration	Validate the current specification against the shared (Pydantic) schema.
	finalize_configuration	Close the elicitation loop once the specification is complete and executable.
Experiment configuration (2, E)	read_baseline	Return the baseline objective and decision-variable parameters.
	propose_experiment	Add one experiment that varies a single factor of the baseline.
	list_experiments	Return all experiments committed so far, in order.
	remove_experiment	Remove a previously proposed experiment by name.
	commit_experiments	Close the design loop and emit the final experiment set.
Running (3, R)	run_optimization	Execute one committed experiment exactly once and return a result summary.
	finalize_results	Write results and figure metadata into the shared state for the Reporter.

parameters, the experiment design when applicable, the execution outputs including results and figure metadata, and the final report. Every node reads from and writes to this state irrespective of whether it operates in WM or RM modes, ensuring that each stage always acts on the most recent and internally consistent view of the task.

On top of the shared state, each node receives a stage-specific prompt assembled only from the subset of state entries relevant to its function: the Planner receives the user query (e.g., the request to minimise pump energy over a 24-hour schedule), the Parameter Elicitor the plan and current specification (e.g., the objective and decision variables agreed so far), the Experiment Configurator the committed baseline configuration (e.g., the finalised optimisation algorithm set-up to be varied into what-if cases), and the Reporter the execution outputs and generated figures (e.g., the finalised GA or NSGA-II set-up to be varied into what-if cases). In WM, this prompt frames a single constrained LLM call; in RM, the same state entries and prompt mappings seed the agent's reasoning loop, and the tools listed in Table 1 read and write the same state. Confining every prompt to a bounded, explicitly mapped subset of the shared state keeps prompts focused, provides a traceable record of how a task evolves from natural language to executable configuration to interpreted results, and allows any node to be switched between modes or replaced without altering the others.

2.4. Performance evaluation

Two sets of metrics are used to evaluate the optimisation pipeline. Solution quality is assessed using the best objective value attained for single-objective tasks and the hypervolume (HV) of the Pareto front for multi-objective tasks (Zitzler and Thiele, 1999), defined as the volume of the objective space dominated by the front and bounded by an analysis specific reference point set strictly beyond the worst-case values observed across all runs on all objectives, such that a larger hypervolume indicates a better-distributed front closer to the true optimum; in the cross-model analysis (Section 4.2), both metrics are expressed as percentages of their respective grand means across all completed runs. The

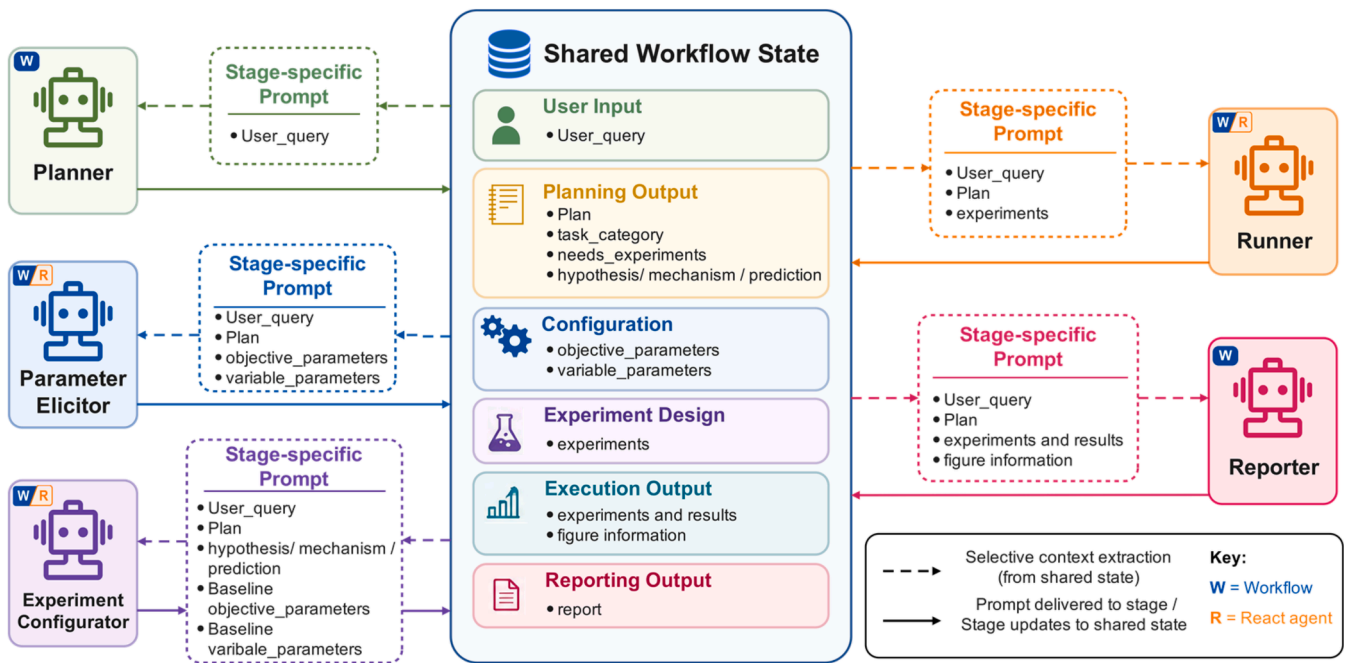


Fig. 2. Shared-state representation and stage-specific prompt construction. The shared workflow state acts as the single source of truth: each node draws a prompt from only the entries relevant to its function and writes its output back to the state, in both workflow and agent modes.

operational cost and reliability of each control architecture are characterised by three metrics recorded at every node: (1) **the end-to-end completion rate**, defined as the fraction of repeated runs that are counted as complete, where a run is complete only if the workflow terminates without an unhandled exception, no node records a failure, and the outputs required by the task are produced, comprising the experiment results and the required objective-related figures, while a run that does not satisfy all of these conditions is counted as failed; (2) **the token consumption and its monetary cost**, where tokens are the basic units of text processed by an LLM and each call separates new input tokens, of which cached input tokens retrieved from the API cache are a subset (Anthropic, 2024), and model generated output tokens. Total token use is the sum of input and output tokens, and cached input tokens are therefore not added separately, whereas cost is obtained by pricing new input, cached input and output tokens at their separate provider rates (OpenAI et al., 2024). Both total token use and monetary cost are therefore adopted as evaluation metrics; and (3) **the number of LLM and tool calls**, which track the API round trips and tool invocations. All three metrics are recorded for every attempted run, whether or not it completed, so that resources spent without returning a result are included. Completion rates are then compared between architectures as model-stratified odds ratios relative to the fully workflow-driven baseline, with profile-likelihood confidence intervals and Holm-adjusted P values (Fischer and Lewis, 2021; Menyhart et al., 2021).

2.5. Algorithm catalogue, architectures, LLMs, and implementation settings

The optimisation tasks defined in Section 2.1 are solved using a Genetic Algorithm (GA) for the single-objective task and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for the multi-objective task, both implemented in the Python library *pymoo* (Blank and Deb, 2020). The non-negative pressure constraint is enforced as a hard constraint through *pymoo*'s default feasibility-first rule, which always ranks feasible solutions above infeasible ones and, when comparing two infeasible solutions, ranks by degree of constraint violation (Blank and Deb, 2020). Additional algorithms available in *pymoo* can also be incorporated by registering them in the configuration

template, without modifying the workflow logic.

In the implementation experiments, five control architectures were evaluated: WWW, RWW, WRW, WWR and RRR. The three letters correspond, in order, to the Parameter Elicitor, Experiment Configurator and Runner, where W denotes WM and R denotes RM. WWW is the fully WM baseline, RWW, WRW and WWR delegate one execution-critical node at a time to RM, and RRR represents full delegation. Because all nodes communicate through the same validated shared state, a downstream node receives a schema-consistent task representation once the preceding node has completed, regardless of whether that state was produced by WM or RM. The selected architectures therefore isolate single-node delegation effects while retaining RRR to capture the cumulative effect of full autonomy.

LLMs are used in this study as text-based reasoning and interaction components embedded in the optimisation pipeline. For the per-task demonstrations in Section 4.1, the text-based nodes use DeepSeek-V4, selected for its stable performance in structured optimisation-related dialogue (DeepSeek-AI, 2026), and the Reporter uses the vision-language model Qwen3-VL to interpret the optimisation plots produced during execution (Bai et al., 2023). To examine whether the effects of control allocation generalise across models of different capability in Section 4.2, all five pipeline nodes are driven by four Qwen3 models (Yang et al., 2025). A single model family is used for this ladder so that model scale varies while the training recipe and tool-calling interface remain constant. The four checkpoints span roughly an order of magnitude in scale (Qwen3-235B, i.e., with 235B total parameters, Qwen3-32B, Qwen3-14B and Qwen3-8B, with only 8B parameters), with the workflow structure, prompts, tools and configuration template held constant. All models belong to open-weight families and are accessed through their official hosted APIs, so the underlying checkpoints remain publicly available for local reproduction. The exact API model identifiers, providers and access dates are listed in Table S2. The temperature parameter controls the randomness of token sampling during LLM decoding, with lower values reducing sampling variability and higher values increasing output diversity (Holtzman et al., 2019). In this study, the temperature is fixed at 0 across all models to reduce stochastic variation and improve run-to-run consistency.

Each pipeline stage is assigned a maximum token budget, which

bounds the length of both input and output text and thereby limits the cost, latency and reasoning depth of each LLM call (Snell et al., 2024). The planning stage is allocated 3,000 tokens to accommodate structured task descriptions and experimental configurations; the reporting stage 1,000 tokens for concise multimodal summarisation; and routine intermediate stages 500 tokens, as they involve shorter structured exchanges. These values were determined empirically to provide sufficient capacity for stage-specific requirements while avoiding unnecessary computational overhead. Where a stage output reaches its budget, as occurs at the reporting stage, an automatic continuation call completes and concatenates the output, so that no content is lost. The budgets did not by themselves cause failures of individual models or affect solution quality. The system messages, human messages and tool call structures used at each stage are provided in Tables S3–S5.

Workflow orchestration is implemented with *LangChain 1.0*, which defines the staged execution structure, shared-state propagation and prompt-based coordination described in Sections 2.2 and 2.3, and provides the RM runtime through LangGraph's prebuilt ReAct agent (LangChain Developers, 2024). Tools are registered through the LangChain tool interface and exposed to the model as JSON schemas through the native tool-calling interface, also known as function calling, of each provider API. Structured outputs at every LLM node are validated against *Pydantic* schemas, which verify each configuration parameter against predefined type, range and value constraints before it reaches the deterministic execution layer, so that malformed or incomplete configurations are caught rather than silently propagated. The validated configuration is then compiled into a *pymoo* problem. Each candidate decision vector is applied to a copy of the WNTR (Kluse et al., 2020) network model and evaluated with the EPANET solver (Rossman et al., 2020), and a failed simulation returns a penalty objective value together with a constraint violation, so that an unsolvable candidate cannot be scored as feasible. All model clients use a 300 s API timeout, and any agent loop that reaches the recursion limit of 50 steps is terminated and recorded as a structured failure. The overall implementation uses Python 3.11 and runs on a fixed platform with an Intel i7-14700HX CPU and an NVIDIA RTX 4060 GPU (8 GB) to ensure consistency across experiments. The complete source code is publicly available on the [GitHub](#) to support transparency and reproducibility.

3. Case study

To evaluate the application of an LLM-driven workflow to WDNs optimisation in a realistic and reproducible setting, this study adopts the C-Town benchmark WDN. C-Town is a hypothetical medium-sized water supply system originally released as part of the Battle of the Water Networks II benchmark initiative (Marchi et al., 2013). The network comprises 388 nodes, 444 pipes, one reservoir, seven tanks, 11 pumps, and four valves, with a total pipe length of approximately 56.7 km, pipe diameters ranging from 51 to 610 mm, and Hazen-Williams roughness coefficients ranging from 60 to 140 (Marchi et al., 2013); the network layout is provided in Figure S1. The network is supplied in EPANET-compatible .inp format, and EPANET serves as the hydraulic simulation engine within the optimisation process. Each candidate solution generated by the workflow is hydraulically evaluated by the EPANET model under the default extended-period simulation settings of the released .inp file (168 h horizon, 1 h hydraulic time step), and the resulting pressures, flows, tank levels, and energy-related variables are used to compute objective values and assess constraint satisfaction.

4. Results

4.1. Workflow and agent control on two representative WDN optimisation tasks

This section examines whether moving control from the human to the LLM changes what the optimisation pipeline produces on well-

specified tasks. Two operational questions were posed on the C-Town network: a single-objective pump energy minimisation and a multi-objective pump energy-MRI optimisation. The optimisation problems were solved using DeepSeek-V4 across the full spectrum between WWW and RRR, with the optimisation pipeline expected to formulate each problem, elicit any missing configuration inputs from the user, design a structured comparison and report the findings. Each run contained a fixed baseline case and two under-specified what-if cases; Sections 4.1.1 and 4.1.2 report these two parts separately.

4.1.1. Both control modes produce identical baseline optimisations

This section examines how optimisation pipeline behaviour compares across the two control extremes, WWW and RRR, on both the single-objective pump energy minimisation task and the multi-objective pump energy-MRI task. In both cases, the user provides a natural-language task description (Table S6) together with the baseline optimisation configuration defined in Table 2: GA with a population of 50 over 50 generations for the single-objective task and NSGA-II with the same budget for the multi-objective task, both with pump-speed multipliers bounded to [0.5, 1.5] and a minimum service pressure of 20 m.

Across both modes, the optimisation pipeline carried each request through the same five stages, as shown in Fig. 3. The Planner expanded the request into a structured plan specifying the objectives, decision variables, algorithm and termination criteria. The Parameter Elicitor then queried the user for any missing configuration inputs, e.g., population size; the user responded with the values defined in Table 2, completing the specification. The Experiment Configurator autonomously designed the two what-if cases around the baseline (Section 4.1.2), the Runner executed each experiment on the EPANET-based hydraulic solver (Rossman et al., 2020), and the Reporter produced a structured interpretation. Under WWW, each of these stages followed a predefined sequence of LLM calls with fixed scope and deterministic hand-offs between sequential stages. Under RRR, each configurable node was executed by a ReAct agent operating over the bounded tool set of Table 1; for example, the Parameter Elicitor agent iteratively called the query, write and validate tools in a loop until all specification fields passed schema validation, before calling the finalise tool, with the Experiment Configurator and Runner agents following analogous tool-calling sequences for their respective tasks. The Runner produced a convergence plot for the single-objective task, showing steady energy reduction across generations, and a Pareto front for the multi-objective task, showing a well-distributed trade-off between pump energy and system resilience; for the baseline these results coincide exactly between the WWW and RRR modes (Fig. 3). The full plans, elicitation traces, configurations and reports for both modes are provided in the Supplementary Information, where a check confirmed that the numerical content was carried through correctly and the figure trends read correctly, that the recommendations were broadly sound but lacked actionable specificity, and that the free-form narrative remains subject to the hallucination inherent to current language models, so expert

Table 2

User-specified baseline configuration for the single- and multi-objective tasks. The engineer fixes every baseline value.

Parameter	Single-objective (specified)	Multi-objective (specified)
Objective(s)	Minimise pump energy	Minimise pump energy; Maximise MRI
Algorithm	GA	NSGA-II
Decision variables	Speed multiplier of each pump at each time step	
Speed multiplier bounds	[0.5, 1.5]	
Population size	50	
Generations	50	
Minimum service pressure	20 m	

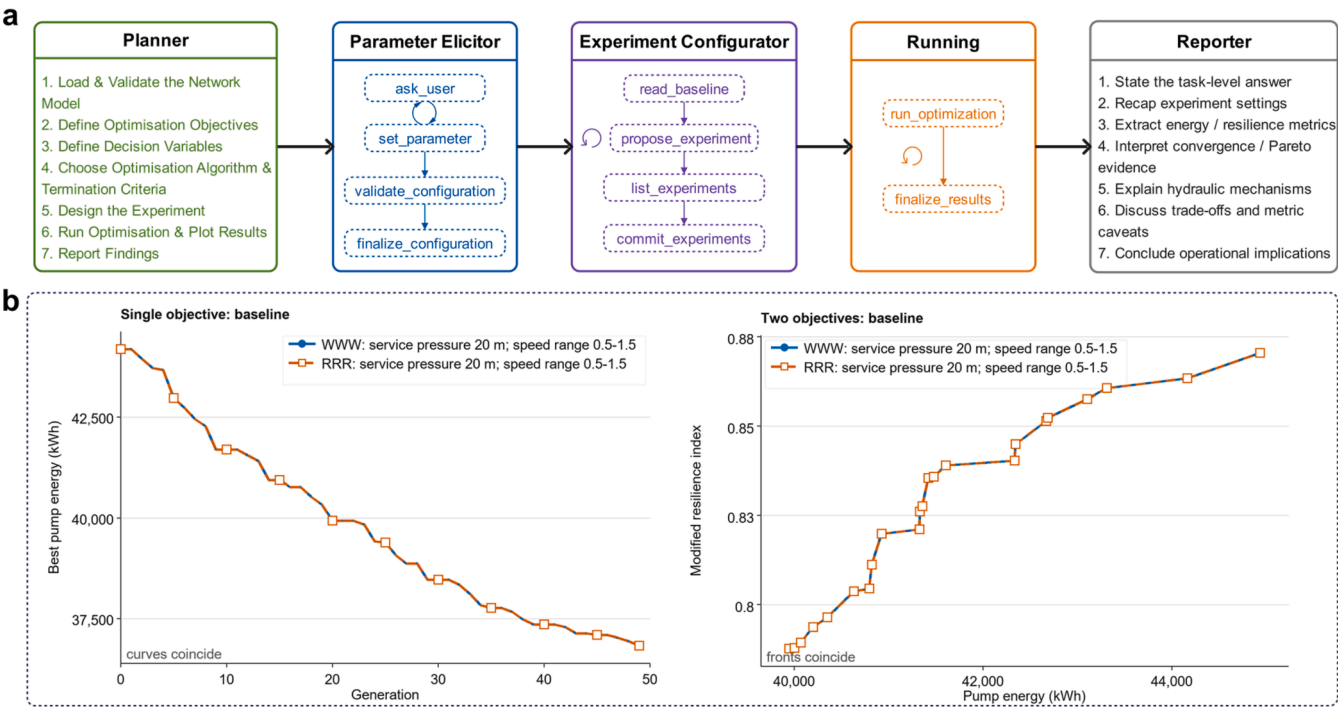


Fig. 3. Optimisation pipeline execution and baseline optimisation outputs. (a) Five-stage WDN optimisation pipeline, with ReAct-mode tool loops shown for configurable nodes. (b) Fixed-baseline optimisation outputs under WWW and RRR, showing coincident single-objective convergence curves and multi-objective Pareto fronts.

judgement is still required.

4.1.2. Under-specified instructions enable exploratory what-if design

The prompt used in this experiment contained both a fixed baseline setting and two compact what-if cases (Table S6). [Section 4.1.1](#)

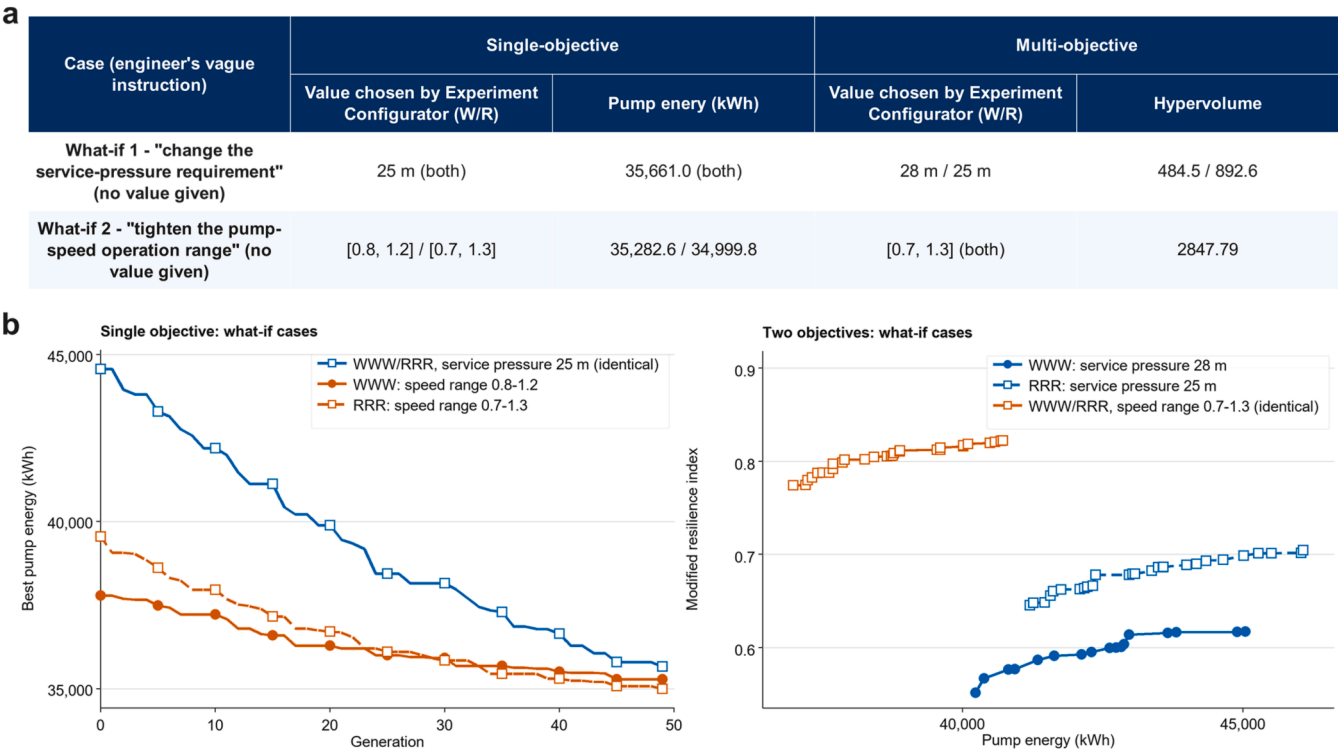


Fig. 4. What-if design choices and optimisation outcomes under under-specified instructions. (a) Values selected by the Experiment Configurator, with the corresponding pump-energy and hypervolume metrics. (b) Single-objective convergence curves and multi-objective Pareto fronts for the same what-if cases. Coincident WWW/RRR series are plotted once; the underlying values are provided in Tables S7 and S8.

examined the fixed-baseline outputs from these runs, whereas this section examines the what-if cases. For the what-if cases, the prompt specified the type of modification, such as changing the service-pressure requirement or tightening the pump-speed operating range, but did not prescribe the numerical values (Fig. 4). The Experiment Configurator therefore instantiated concrete scenario values from the same task prompt, providing a controlled form of exploratory scenario design within the optimisation pipeline.

As shown in Fig. 4, the selected what-if values were identical in some cases and different in others. For the single-objective task, both modes selected 25 m for the service-pressure case, but WWW tightened the pump-speed range to [0.8, 1.2], whereas RRR selected [0.7, 1.3]. For the multi-objective task, both modes selected [0.7, 1.3] for the pump-speed case, but WWW raised the service-pressure requirement to 28 m, whereas RRR selected 25 m. This mixed pattern indicates that the specific what-if values reflect LLM-side scenario instantiation rather than a systematic effect of WM or RM. Once the scenario values were fixed, deterministic execution governed the optimisation outputs: identical values produced identical results, while different values produced different magnitudes but consistent engineering trends (Fig. 4). Thus, the useful freedom lies in scenario generation, not in changing solver behaviour.

4.1.3. The cost of autonomy: token consumption and model calls

If increased autonomy does not change answers from those obtained by the human-defined workflow, its cost must be measured through the resources required to reach it. Following the token definitions and cost decomposition of Section 2.4, each node was instrumented to record token consumption, LLM call counts and tool call counts (such as EPA-NET runs) at every invocation, enabling a fair resource comparison between WWW and RRR across both tasks.

As shown in Fig. 5, the two modes consumed substantially different resources to reach the same answer. Token consumption (Fig. 5) was

dominated by the Parameter Elicitor in both modes and both tasks, reflecting its iterative dialogue with the user. Under WWW, preceding exchanges accumulate as cached context, keeping new input tokens moderate, but consumption varies with task complexity: 39.2k for the single-objective task and 77.0k for the multi-objective task. Under RRR, each tool call generates additional input and output tokens that cannot be cached across calls, producing a larger absolute footprint (54.6k and 77.0k, respectively), but the increase from single- to multi-objective task is proportionally smaller, suggesting that agent-mode consumption is less sensitive to task complexity than workflow-mode consumption. The Running node consumed negligible tokens under WWW because the solver was invoked directly without any LLM call once the configuration was finalised. Call counts differed even more sharply (Fig. 5): the Parameter Elicitor required 3 LLM calls under WWW but 17 LLM calls and 26 tool calls under RRR for the single-objective task, while the Planner and Reporter remained unchanged at 2 calls each. The cost of autonomy is therefore concentrated at the formulation nodes and scales with the number of nodes delegated to agent control, a relationship quantified across model scales in Section 4.2.

4.2. Transferring control across model capability

The per-task comparison in Section 4.1 fixes the model and contrasts the two extreme architectures. To characterise how delegating each node affects reliability and cost across models of differing capability, a controlled experiment was conducted across all five architectures (WWW, RWW, WRW, WWR and RRR) and five LLMs spanning roughly an order of magnitude in parameter scale (DeepSeek-V4 and Qwen3-235B, -32B, -14B and -8B), with ten repetitions per combination, giving 250 end-to-end runs in total. To support batch evaluation, each run used a combined task in which the Experiment Configurator, following the task description in Table S6, takes a single-objective baseline and expands it into a matched multi-objective comparison, so that both tasks

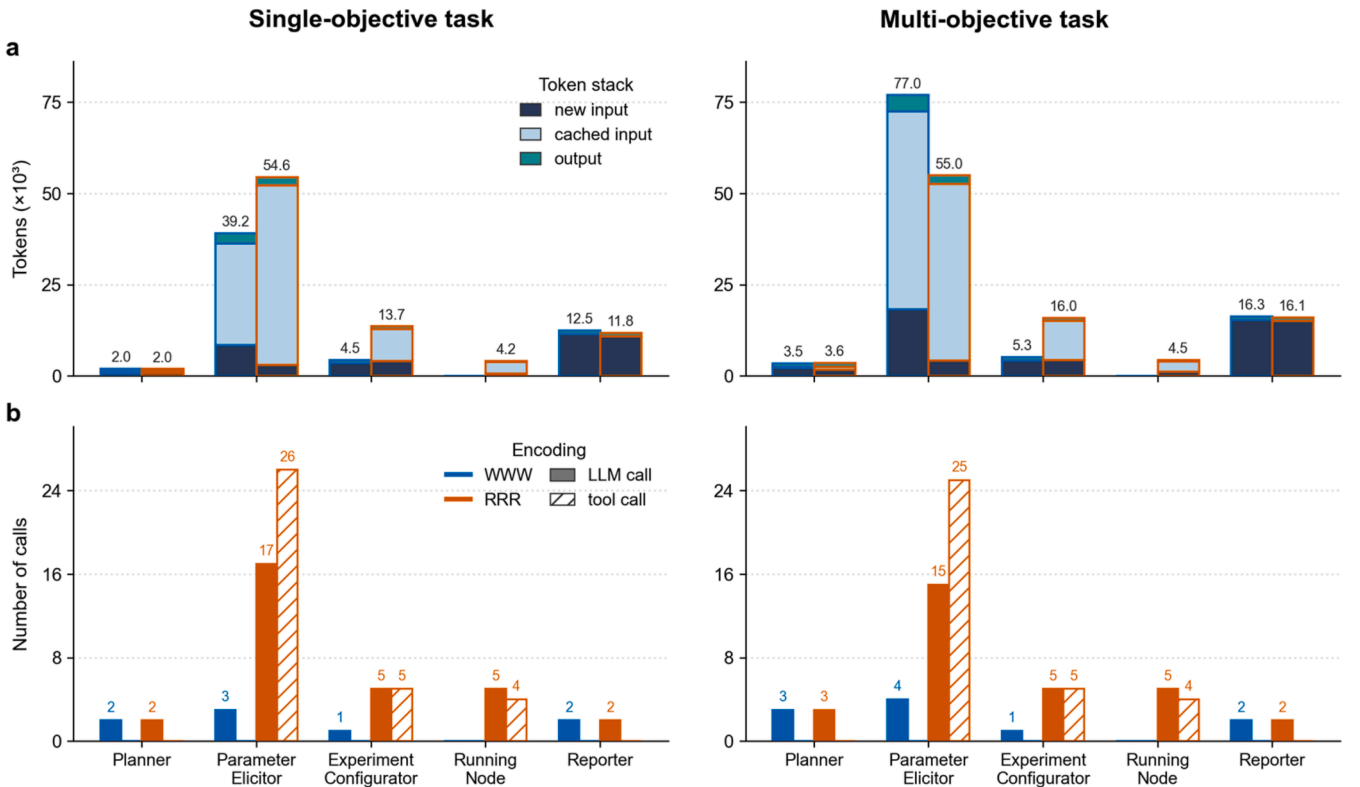


Fig. 5. Resource cost of control allocation under DeepSeek-V4. (a) Per-node token consumption (new input, cached input and output) and (b) per-node LLM-call and tool-call counts, for the fully workflow-driven (WWW) and fully agentic (RRR) architectures on the single- and multi-objective tasks.

are executed within a single pipeline run. Population size and number of generations were set to 5 each, to reduce computational cost at scale, with all other settings consistent with Table 2. Holding the task, tools and prompts constant isolates two experimental variables: the degree to which control has been shifted to the LLM, and the capability of that LLM.

4.2.1. Solution quality is reproducible regardless of where control sits

Following the evaluation protocol of Section 2.4, completion rate was calculated across the 10 repeated runs for each architecture-model combination. Solution quality was assessed only for completed runs, with the two metrics expressed as percentages of their respective grand means across all completed runs. Of the 250 runs in the full experiment, 158 completed successfully and were retained for the quality analysis. The pump energy and hypervolume obtained in each of these runs are listed in Tables S9 and S10, where failed runs are marked as such.

As shown in Fig. 6, completion rate and solution quality responded very differently to control allocation and model scale. Completion rate (Fig. 6) depended strongly on both axes. DeepSeek-V4 completed all runs across every architecture. Among the Qwen3 models, completion declined with increasing autonomy and decreasing model scale, though not monotonically: even Qwen3-235B showed partial failures under certain architectures. Pooled over the fifty runs that each model contributes, completion ordered with model capability, at 86%, 70% and 60% for Qwen3-235B, Qwen3-32B and Qwen3-14B. Because the optimisation pipeline prompts and tool interfaces were developed and tuned on DeepSeek-V4, the overall pattern indicates that completion reliability under agent control depends on prompt-model compatibility as much as on raw model capability, with specifications optimised for one model transferring imperfectly to others. The parameter node was the most failure-prone point, as sustaining a coherent multi-turn dialogue with the user to elicit all required inputs demands consistent instruction-following across many sequential steps. Qwen3-8B completed no runs under any architecture in this setup, indicating a practical capability threshold for agent-driven formulation. In contrast, solution quality among completed runs was largely invariant to control allocation (Fig. 6): the within-architecture coefficient of variation was 1.2% for pump energy and 12.9% for hypervolume, with architecture means spanning 100% and 97–101% of the grand mean, respectively, and no architecture showing a systematic advantage. Replicated across five architectures and five models of varying scale, this corroborates the conclusion drawn in Section 4.1.1 from the WWW-RRR comparison: the optimisation outcome is governed by the algorithm and configuration, not by where control resides. When a run succeeds, autonomy therefore

raises the probability of failure without improving the answer.

4.2.2. Failure is structured and concentrated at the delegated node

When an autonomous run failed the completion criterion of Section 2.4, the condition on which it terminated was identified from the recorded execution trace. Each failed run was assigned to one of three structured forms (Fig. 7a). In over-asking, the run terminated with the specification still requiring user input. The parameter agent repeatedly queried the user for information that should have been inferred from earlier responses or task defaults, trapping the node in a clarification loop that never converged. In incomplete deliverables, every node returned but the required outputs were absent. The task specified more than one experiment, but a node committed a partial specification and ran only the baseline, omitting the planned comparison, so the run lacked the experiments needed to answer the task and produced fewer objective-related figures than the task required. In invalid configuration, a node rejected the specification as unsupported. The experiment agent emitted a specification that violated the configuration schema of Table S1, such as pairing a single-objective algorithm with a multi-objective request. Because execution halts at the first blocking condition, each failed run carries a single recorded termination condition. One complete execution trace for each mode is provided in the Supplementary Information.

As shown in Fig. 7b, each failure mode mapped onto the node that had been delegated to the agent. Incomplete deliverables were the most common mode: the combined task requires the agent to expand a single-objective baseline into a matched multi-objective comparison, which involves coordinated changes to several coupled fields, including the algorithm and objective set, and weaker models frequently committed an experiment set before all the required changes were complete. This dominated the failures of Qwen3-32B under parameter delegation (RWW). Over-asking appeared specifically when the parameter node was delegated: whereas the workflow path advances through a fixed elicitation sequence, the agent path must judge for itself when the specification is complete, and weaker models failed to recognise that the necessary information was already available, accounting for all RWW failures of Qwen3-14B. Delegating the experiment node (WRW) instead produced invalid configurations, as seen in Qwen3-32B, while delegating the running node (WWR), the most constrained of the three, was tolerated by every model that completed at all. Qwen3-8B is the limiting case, completing no run under any architecture: it failed through incomplete deliverables wherever the parameter node remained a workflow step and through over-asking wherever it was delegated. Full delegation (RRR) introduced no new failure mode but inherited the

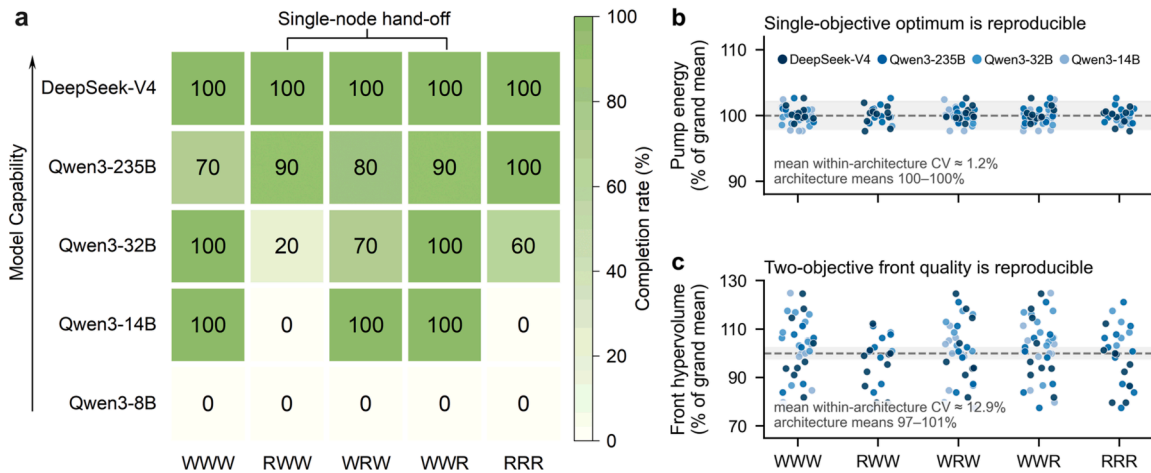


Fig. 6. Completion and solution quality across the control-transfer landscape. (a) End-to-end completion rate (%) for five architectures (columns) and five LLMs (rows), ordered by capability. (b) Single-objective optimal pump energy and (c) multi-objective Pareto-front hypervolume on completed runs, each expressed as a percentage of its respective grand mean across all completed runs.

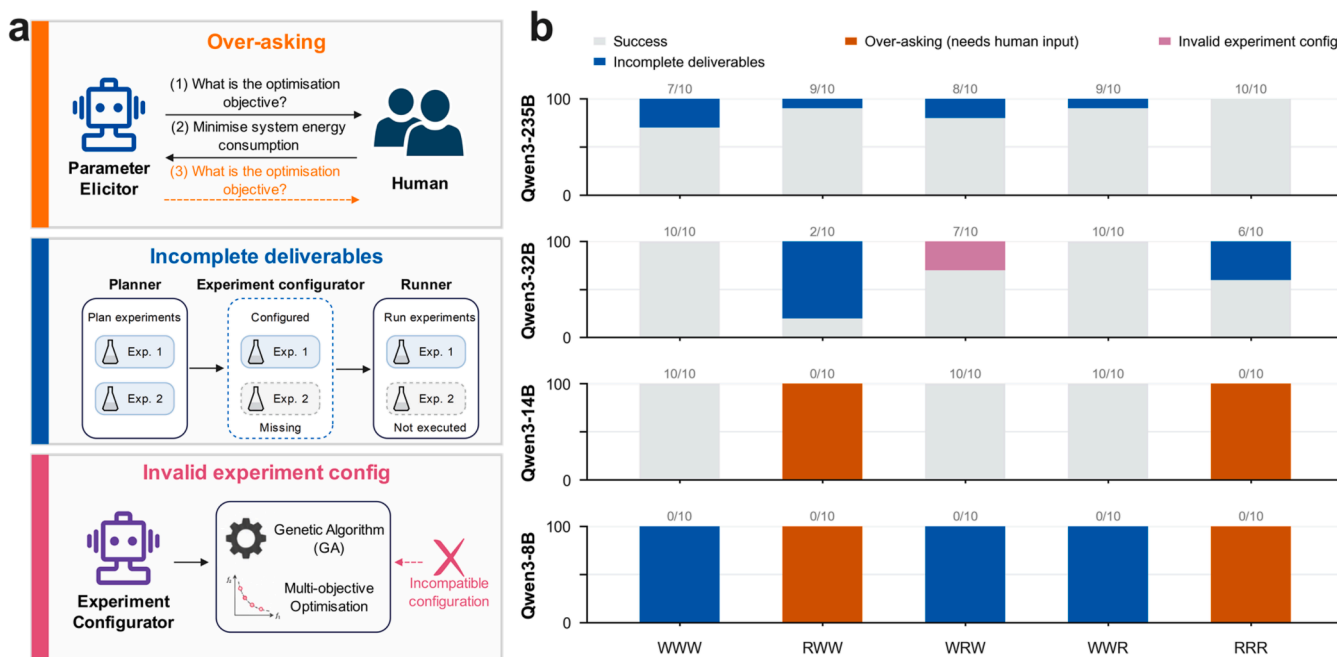


Fig. 7. The structure of failure under delegated control. (a) The three recurring failure modes: over-asking at the parameter node, incomplete deliverables along the optimisation pipeline, and invalid experiment configurations. (b) Outcome of ten repetitions for each of the four Qwen3 models under the workflow baseline (WWW), the three single-node delegations (RWW, WRW, WWR) and full delegation (RRR), with each bar decomposed into successful runs and the three failure modes.

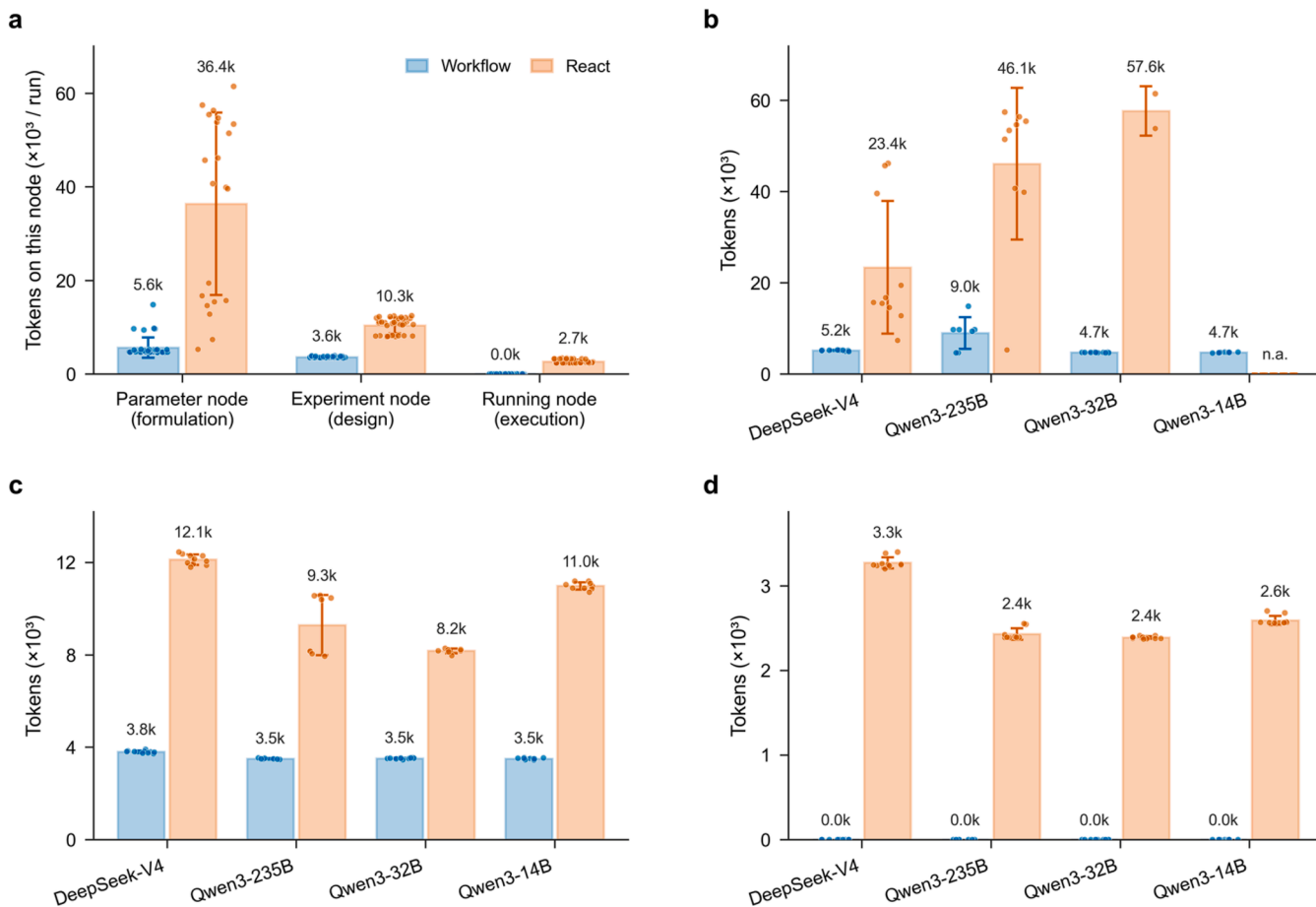


Fig. 8. The token cost of delegated control. (a) Tokens consumed per run at the parameter, experiment and running nodes in workflow versus ReAct-agent mode, pooled across models. (b-d) The same comparison resolved by model for the parameter, experiment and running nodes. Bars and labels show mean token use, points show individual successful runs, error bars show mean $\pm$ s.d., and the workflow running node uses no tokens by design.

signature of its most fragile node: Qwen3-235B sustained full autonomy and completed all ten runs, whereas Qwen3-32B failed through incomplete deliverables and Qwen3-14B through over-asking, the same modes seen when their parameter node alone was delegated. Failure is therefore not random but tied to the locus of control: each delegated node introduces a characteristic failure at that node, the parameter node is the binding constraint under full autonomy, and the capability required to avoid failure rises with the degree of autonomy granted.

4.2.3. The autonomy token cost grows with each delegated node

To quantify the resource cost of transferring control between the human-defined workflow and the LLM agent, token consumption was recorded per node and per run on successfully completed runs only, measured separately in workflow and ReAct-agent mode. The per-node analysis of Fig. 8 is restricted to successfully completed runs, so that node-level costs are comparable across architectures; the whole-run analysis of Table 3 additionally includes every attempt that failed, since resources consumed without returning a result form part of the operational cost of an architecture. Because the Running node executes no LLM call under workflow mode, its workflow cost is zero by design, providing a clean baseline against which delegation costs can be read directly.

As shown in Fig. 8, delegation increased token consumption at every node, with the increase concentrated where reasoning demands are highest. Aggregated across all models (Fig. 8), delegating the parameter node raised its token consumption from approximately 6,000 to 36,000 tokens per run; the experiment node increased from approximately 4,000 to 10,000 tokens; and the running node rose from zero to approximately 3,000 tokens. The same pattern held across individual models (Fig. 8), though with a notable difference in variability: under DeepSeek-V4, agent-mode consumption at the parameter node was relatively stable, reflecting consistent and decisive tool-calling behaviour; under smaller Qwen3 models, consumption was substantially higher and more variable, as weaker models required more iterations to converge on a complete specification. Across all models, agent-mode consumption at the parameter node was five to twelve times its workflow counterpart. Since these costs accumulate across nodes, the fully agentic architecture incurs a compounding token overhead at every stage of the optimisation pipeline. As demonstrated in Sections 4.1.1 and 4.2.1, this overhead produces no difference in optimisation outcomes relative to workflow mode. Delegating control to the LLM therefore imposes a structured and cumulative token cost that scales with the number of nodes delegated, without any corresponding improvement in the answer.

Fig. 8 resolves this cost by node for completed runs, whereas Table 3 reports whole-run outcomes separately for completed and failed attempts. Two results follow. First, a statistically detectable reduction in completion appeared only in architectures in which the formulation node was delegated: relative to WWW, the odds of completion were lower for RWW (OR 0.056, 95% CI 0.010–0.216, Holm-adjusted $P < 0.001$) and RRR (OR 0.097, 95% CI 0.017–0.396, Holm-adjusted $P = 0.006$); WRW and WWR did not differ detectably from WWW. Second, failure was not cheap in architectures with parameter-node delegation. Under RWW and RRR, failed runs consumed 62.8k and

70.2k tokens and cost 20.1×10^{-3} and 22.2×10^{-3} US\$ per run, respectively, exceeding the corresponding successful-run means and the successful-run means of all workflow-dominant architectures. This pattern is consistent with the failure analysis in Section 4.2.2, particularly over-asking at the delegated parameter node, where repeated clarification turns can accumulate without yielding a complete specification.

5. Discussion

5.1. Extensibility of the configurable workflow-agent architecture

The configurable architecture studied here separates two concerns that are often conflated in LLM-driven automation: (1) what the optimisation pipeline does, and (2) who decides how it does it. Because these concerns are decoupled, the pipeline's value does not rest on any particular algorithm or benchmark, and new objectives, decision variables, constraints, solvers or analysis steps can be incorporated through the canonical configuration template without altering the control logic. The same framework can therefore serve the broad range of WDN operational and planning problems, including design, rehabilitation, pressure management and uncertainty-aware scheduling, that are today implemented as separate bespoke scripts (Mala-Jetmarova et al., 2017; Sarbu, 2021). Because the LLM-driven nodes are cleanly separated from the deterministic execution layer, the architecture is also not domain-specific: substituting the simulator, for example the Storm Water Management Model for urban drainage (Rossman, 2022), would carry the same control structure to flood mitigation, stormwater management or integrated water-energy planning while leaving the interaction, configuration and reporting logic intact (Dobson et al., 2025; Li et al., 2024). Together, the swappable execution layer, shared configuration template and decoupled control logic make the architecture a reusable basis for LLM-assisted engineering optimisation that can be adapted to new tasks, domains and solvers with minimal redesign effort.

5.2. How much control should be delegated to the LLM?

The results presented in Sections 4.1 and 4.2 reveal an asymmetry at the heart of control allocation: for well-specified optimisation tasks, the locus of control governs the reliability and cost of automation far more than it governs solution quality. Quality was effectively invariant across architectures, with optimal energy and front hypervolume varying by only a few percent and no systematic advantage to autonomy (Sections 4.1.1 and 4.2.1). What delegation altered was the probability of run completion and the computational resources consumed. The reason is that autonomy converts a deterministic, auditable procedure into a stochastic one: even at zero temperature, an agent's freedom to select its next action allows minor variations in intermediate text to propagate into different tool calls and, occasionally, into the structured failure modes described in Section 4.2.2 (Qiu et al., 2025). For water infrastructure analysis this has substantive implications, as decisions are frequently subject to audit, compared across scenarios and revisited over long time horizons under the assumption that identical inputs yield identical analyses; a

Table 3
End-to-end reliability and per-run operational cost across control architectures.

Architecture	Completion OR vs WWW (95% CI); Holm-adjusted P	Successful tokens (k)	Successful cost ($\times 10^{-3}$ US\$)	Failed tokens (k)	Failed cost ($\times 10^{-3}$ US\$)
WWW	Reference	11.2 $\pm$ 2.1	2.853 $\pm$ 1.797	8.7 $\pm$ 2.2	1.542 $\pm$ 1.356
RWW	0.056 (0.010–0.216); $P < 0.001$	42.2 $\pm$ 19.4	9.765 $\pm$ 8.812	62.8 $\pm$ 3.5	20.098 $\pm$ 1.042
WRW	0.541 (0.100–2.518); $P > 0.99$	17.7 $\pm$ 2.3	3.817 $\pm$ 2.295	9.2 $\pm$ 3.4	1.973 $\pm$ 2.033
WWR	3.611 (0.391–79.183); $P > 0.99$	14.1 $\pm$ 2.3	3.571 $\pm$ 2.152	10.6 $\pm$ 3.1	1.468 $\pm$ 1.567
RRR	0.097 (0.017–0.396); $P = 0.006$	51.7 $\pm$ 23.2	13.644 $\pm$ 10.369	70.2 $\pm$ 3.6	22.181 $\pm$ 1.010

Each architecture comprised 50 runs. Token use and API costs are reported as mean $\pm$ s.d. per run, separately for completed and failed runs. Completion effects are conditional common ORs stratified by model and estimated relative to WWW. Profile-likelihood 95% CIs were calculated for the completion ORs (Fischer and Lewis, 2021), and exact conditional P values for the four prespecified comparisons were adjusted using Holm's procedure (Menyhart et al., 2021).

human-defined workflow preserves that assumption by construction, whereas an agent operating without prescribed steps does not (Anthropic, 2024; Yu et al., 2025). The behaviour of such an agent cannot be established from a single execution and must instead be assessed over repeated runs. Taken together, the results reveal a consistent reliability-cost trade-off: greater LLM autonomy increases resource consumption while reducing execution reliability, despite having little influence on optimisation quality.

These observations do not suggest that workflow mode is universally preferable, but rather that the appropriate degree of delegation depends on where in the optimisation pipeline it is applied. LLM autonomy is well suited to the interface, including understanding intent, planning at a high level and communicating results, whereas a predefined workflow is better suited to preserving consistency and reproducibility at the execution layer (Li et al., 2024; Yu et al., 2025). The failure and cost patterns observed across nodes reinforce this distinction: formulation nodes, which require sustained multi-turn reasoning to produce a complete and valid specification, carry disproportionate failure risk under agent control, whereas execution nodes with well-constrained tool interfaces proved robust even in smaller models. Completion therefore depends not only on model scale but also on how well a model fits the prompts and tool interfaces it is given, which makes the locus of delegation a more transferable design choice than the choice of model. As WDN problems grow to encompass many scenarios and operating conditions, a single fixed workflow may not suffice, and a hierarchical arrangement becomes attractive in which an agent at the top selects or composes validated workflows while execution within each workflow remains deterministic (Fu, 2025; Masters et al., 2025).

5.3. Efficiency, risk and accessibility for water utilities

The practical implications of control allocation extend beyond optimisation pipeline performance to the conditions under which automated optimisation is viable for water utilities. The autonomy tax of Section 4.2.3, a sixfold increase in tokens at the formulation node and an order-of-magnitude increase in model calls, translates into higher inference cost and longer end-to-end latency. Unsuccessful attempts consume substantial resources and, when the parameter node is delegated, can cost more than successful runs, so the cost of obtaining an answer rises faster than the cost of running the pipeline once. Reliability and cost are therefore not independent design criteria, since the same delegation that lowers the chance of completion also raises the price of each attempt. For routine, well-specified analyses this overhead accumulates without returning engineering value, though it may be justified where a task genuinely benefits from flexible reasoning. The capability floor of Section 4.2.1 introduces a further consideration: the workflow-dominant architecture completed successfully even on a 14B-parameter model, which makes automated optimisation feasible on smaller, locally hosted models. Local hosting eases two practical concerns for utilities. It replaces per-token charges and dependence on a frontier API with a fixed capacity cost, and it keeps operational network data inside the organisation, since a workflow layer can confine sensitive information to a controlled local environment. Of the benefits claimed for automation, reproducibility is the one quantified in this study. Accessibility and reduced manual effort matter equally for adoption but would require a controlled user study to measure. These considerations suggest that the choice between workflow and agent control should be informed by the operational context, available infrastructure and acceptable level of execution risk, and align naturally with the move towards digital-twin environments where reliable, locally deployable automation is a prerequisite for adoption (Hosseini et al., 2026; Lee et al., 2025).

5.4. Limitations and future directions

The findings of this study should be interpreted in light of several boundary conditions. The tasks are limited to operational pump-speed

control on a single benchmark network and do not span the full diversity of design, rehabilitation and uncertainty-driven problems encountered in practice. The study is also not an algorithm-level evaluation: it uses standard evolutionary solvers to exercise the optimisation pipeline and makes no claim about solver competitiveness. The control axis was applied only to the three execution-critical nodes, with the Planner and Reporter held in workflow mode throughout, so the effects of delegating these boundary nodes remain untested. Report content was likewise verified only for the demonstration runs, and its systematic verification remains an open problem beyond the scope of this study. Finally, the cross-capability results depend on the specific models and API-based inference tested here, which cover one open-weight model family and one independent model, and the observed capability floor should be read as indicative rather than exact, as it will shift with advances in model efficiency and the availability of locally deployable alternatives. Importantly, the appropriate balance between workflow and agent control was established empirically through the control-transfer experiment rather than derived from first principles and may differ for tasks or models beyond those studied.

Future work should address these limitations along four lines. First, extending the framework to a broader range of objectives, constraints and problem classes, with specialised solvers incorporated as interchangeable execution modules, would test whether the control-allocation principle identified here generalises beyond the tasks studied. Second, developing the hierarchical agent-over-workflow arrangement outlined in Section 5.2, in which an upper agent layer composes validated workflows while execution remains deterministic, would allow the balance between flexibility and reproducibility to be managed adaptively rather than fixed in advance (Aquino et al., 2025; Joshi, 2025; Yu et al., 2025). Third, characterising the capability floor across more models and additional open-weight model families, including locally deployable ones, and across more demanding tasks would clarify where automated optimisation is currently feasible and inform deployment in operational settings such as digital-twin environments and larger infrastructure systems, where reproducible, auditable and affordable decision support under evolving conditions is the ultimate objective (Hosseini et al., 2026; Lee et al., 2025). Fourth, the tool interface can itself be treated as an experimental variable. A standardised, schema-first protocol such as the Model Context Protocol could be compared with the LangChain-native interface used here, under identical schemas in both control modes. This would test whether a more structured interface reduces the over-asking observed at the parameter node.

6. Conclusions

This study systematically investigated the delegation of control from deterministic workflows to LLM agents for WDN optimisation through a controlled experimental approach. By re-casting an end-to-end optimisation pipeline as a sequence of nodes that can operate either as deterministic workflow steps or autonomous ReAct-agents, control was progressively transferred node by node across two operational tasks and five LLMs spanning an order of magnitude in model size. The results show that, for well-specified optimisation tasks, the locus of control does not change the engineering results: deterministic workflows and agent based implementations projected comparable configurations and objective values, with solution quality differing by only a few percent across the entire workflow-to-agent spectrum. However, control delegation strongly changes operational reliability and computational efficiency. Granting greater autonomy to the LLM raised token use at the formulation node roughly six fold, multiplied model calls by an order of magnitude, and lowered end-to-end completion rates through a small number of structured failure modes that became more pronounced for smaller LLMs. Overall, increasing LLM autonomy produces little improvement in optimisation quality, but incurs progressively higher computational costs and lower execution reliability, revealing a clear

reliability–cost trade-off associated with control delegation.

Taken together, these findings suggest that, for well-specified WDN optimisation tasks, decision authority should remain with human-defined workflows at the formulation and execution steps, where correctness, reproducibility, and computational efficiency are paramount. LLM autonomy is better suited to open-ended interface-level tasks, such as interpreting user requests, coordinating interactions, and communicating optimisation results. For the water sector, this division of responsibilities enables automated optimisation systems both trustworthy and practical: reproducible and deterministic where engineering rigour is required for infrastructure decisions, and agentic where flexibility and natural language interaction add value. By clarifying where autonomy is best suited, this study offers a principled basis for designing reliable LLM-driven decision support systems for water infrastructure and offers a practical pathway for embedding LLM agents into digital twins and water system operations.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used GPT 5.5 in order to improve language and readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Data availability

All source data, example agent conversation logs (chat transcripts) used for evaluation, and the implementation code that support the findings of this study are available at [GitHub](#).

CRedit authorship contribution statement

Jian Wang: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Shuming Liu:** Writing – review & editing, Supervision. **Guangtao Fu:** Writing – review & editing, Supervision, Funding acquisition. **Dragan Savic:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Jian Wang is a PhD student sponsored by the China Scholarship Council (202408340010) at the University of Exeter. Dragan Savic has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. [951424]). Guangtao Fu has received funding from the British Council under the UK-China Institutional Partnership. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at [doi:10.1016/j.watres.2026.126823](https://doi.org/10.1016/j.watres.2026.126823).

References

Anthropic, 2024. Building effective AI agents [WWW Document]. URL <https://www.anthropic.com/engineering/building-effective-agents> (accessed 4.28.26).

- Aquino, G.D.A.E., Azevedo, N.D.S.D., Okimoto, L.Y.S., Camelo, L.Y.S., Bragança, H.L.D.S., Fernandes, R., Printes, A., Cardoso, F., Gomes, R., Torné, I.G., 2025. From RAG to multi-agent systems: a survey of modern approaches in LLM development. <https://doi.org/10.20944/preprints202502.0406.v1>.
- Awwalu, H.B., Abdullahi, N., Hussaini, M., 2024. Single and multi-objective optimization of water distribution system: a review and bibliographic analysis. *Niger. Def. Acad. J. Mil. Sci. Interdiscip. Stud.* 1.
- Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., Zhou, J., 2023. Qwen-VL: a versatile vision-language model for understanding, localization, text reading, and beyond. <https://doi.org/10.48550/arXiv.2308.12966>.
- Blank, J., Deb, K., 2020. Pymoo: multi-objective optimization in python. *IEEE Access* 8, 89497–89509. <https://doi.org/10.1109/ACCESS.2020.2990567>.
- Brahmbhatt, P., Maheshwari, A., Gudi, R.D., 2023. Digital twin assisted decision support system for quality regulation and leak localization task in large-scale water distribution networks. *Digit. Chem. Eng.* 9, 100127. <https://doi.org/10.1016/j.dche.2023.100127>.
- Cassiolato, G.H.B., Ruiz-Femenia, J.R., Salcedo-Diaz, R., Ravagnani, M.A.S.S., 2024. Water distribution networks optimization considering uncertainties in the demand reading, and beyond. *Water Resour. Manage.* 38, 1479–1495. <https://doi.org/10.1007/s11269-024-03733-y>.
- DeepSeek-AI, 2026. DeepSeek-V4: towards highly efficient million-token context intelligence (technical report). https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/resolve/main/DeepSeek_V4.pdf?download=true.
- Dini, M., Hemmati, M., Hashemi, S., 2022. Optimal operational scheduling of pumps to improve the performance of water distribution networks. *Water Resour. Manage.* 36, 417–432. <https://doi.org/10.1007/s11269-021-03034-8>.
- Dobson, B., Jovanovic, T., Alonso-Álvarez, D., Chegini, T., 2025. SWM many where: a workflow for generation and sensitivity analysis of synthetic urban drainage models, anywhere. *Environ. Model. Softw.* 186, 106358. <https://doi.org/10.1016/j.envsoft.2025.106358>.
- Fischer, S.M., Lewis, M.A., 2021. A robust and efficient algorithm to find profile likelihood confidence intervals. *Stat. Comput.* 31, 38. <https://doi.org/10.1007/s11222-021-10012-y>.
- Fu, G., 2025. Towards autonomous planning and management of urban water systems. <https://doi.org/10.22541/essoar.175745445.50927919/v1>.
- Giustolisi, O., Berardi, L., Laucelli, D., Savic, D., Walski, T., Brunone, B., 2014. Battle of background leakage assessment for water networks (BBLAWN). In: At WDSA Conference 2014. *Procedia Engineering*, 16th Water Distribution System Analysis Conference, WDSA2014, 89, pp. 4–12. <https://doi.org/10.1016/j.proeng.2014.11.153>.
- Goldshtein, Y., Perelman, G., Schuster, A., Ostfeld, A., 2025. Large language models for water distribution systems modeling and decision-making. <https://doi.org/10.48550/arXiv.2503.16191>.
- Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y., 2019. The curious case of neural text degeneration. In: *Presented at the International Conference on Learning Representations*.
- Horsburgh, J.S., Black, S., Castronova, A., Dash, P.K., 2025. Advancing open and reproducible water data science by integrating data analytics with an online data repository. *Environ. Model. Softw.* 188, 106422. <https://doi.org/10.1016/j.envsoft.2025.106422>.
- Hosseini, S.H., Zolghadr-Asli, B., Tenkanen, H., Madani, K., Matin, M.A., Demir, I., Ostfeld, A., Singh, V.P., Savic, D., 2026. Making waves: a conceptual framework exploring how large language model-based multi-agent systems could reshape water engineering. *Water Res.* 291, 125157. <https://doi.org/10.1016/j.watres.2025.125157>.
- Jayaram, N., Srinivasan, K., 2008. Performance-based optimal design and rehabilitation of water distribution networks using life cycle costing. *Water Resour. Res.* 44. <https://doi.org/10.1029/2006WR005316>.
- Joshi, S., 2025. A literature review of Gen AI agents in financial applications: models and implementations. <https://doi.org/10.2139/ssrn.5133985>.
- Klise, K., Hart, D., Bynum, M., Hogge, J., Haxton, T., Murray, R., Burkhardt, J., 2020. Water network tool for resilience (WNTR). User manual, Version 0.2.3. p. SAND-2020-9301R, EPA/600/R-20/185, 1660790. <https://doi.org/10.2172/1660790>.
- LangChain Developers, 2024. LangChain.
- Lee, H.M., Jamali, R., Lazarova-Molnar, S., 2025. A Conceptual framework for digital twins of multi-agent systems. In: *Procedia Computer Science*, The 16th International Conference on Ambient Systems, Networks and Technologies Networks (ANT)/the 8th International Conference on Emerging Data and Industry 4.0 (EDI40), 257, pp. 321–328. <https://doi.org/10.1016/j.procs.2025.03.043>.
- Li, X., Wang, S., Zeng, S., Wu, Y., Yang, Y., 2024. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinityearth* 1, 9. <https://doi.org/10.1007/s44336-024-00009-2>.
- Liu, B., Li, X., Zhang, J., Wang, J., He, T., Hong, S., Liu, H., Zhang, S., Song, K., Zhu, K., Cheng, Y., Wang, S., Wang, X., Luo, Y., Jin, H., Zhang, P., Liu, O., Chen, J., Zhang, H., Yu, Z., Shi, H., Li, B., Wu, D., Teng, F., Jia, X., Xu, J., Xiang, J., Lin, Y., Liu, T., Liu, T., Su, Y., Sun, H., Berseht, G., Nie, J., Foster, I., Ward, L., Wu, Q., Gu, Y., Zhuge, M., Tang, X., Wang, H., You, J., Wang, C., Pei, J., Yang, Q., Qi, X., Wu, C., 2025. Advances and challenges in foundation agents: from brain-inspired intelligence to evolutionary, collaborative, and safe systems. <https://doi.org/10.48550/arXiv.2504.01990>.
- Mala-Jetmarova, H., Sultanova, N., Savic, D., 2017. Lost in optimisation of water distribution systems? A literature review of system operation. *Environ. Model. Softw.* 93, 209–254. <https://doi.org/10.1016/j.envsoft.2017.02.009>.
- Marchi, A., Salomons, E., Ostfeld, A., Kapelan, Z., Simpson, A., Zecchin, A.C., Maier, H. R., Wu, Z., Elsayed, S., Song, Y., Walski, T., Stokes, C., Wu, W., Dandy, G.C., Alvisi,

- S., Creaco, E., Franchini, M., Saldarriaga, J., Páez, D., Hernandez, D., Bohórquez, J., Bent, R., Coffrin, C., Judi, D., McPherson, T., van Hentenryck, P., Matos, J., Monteiro, A., Matias, N., Yoo, D., Lee, H., Kim, J., Iglesias-Rey, P., Martínez-Solano, F., Mora-Meliá, D., Ribelles-Aguilar, J., Guidolin, M., Fu, G., Reed, P.M., Wang, Q., Liu, H., McClymont, K., Johns, M.B., Keedwell, E., Kandiah, V., Jasper, M., Drake, K., Shafiee, E., Barandouzi, M., Berglund, A., Brill, D., Mahinthakumar, G., Ranjithan, R., Zechman, E., Morley, M.S., Tricarico, C., de Marinis, G., Tolson, B., Khedr, A., Asadzadeh, M., 2013. The battle of the water networks II (BWN-II).
- Masters, C., Vellanki, A., Shangguan, J., Kultys, B., Gilmore, J., Moore, A., Albrecht, S., 2025. Orchestrating human-AI teams: the manager agent as a unifying research challenge. In: Proceedings of the 2025 7th International Conference on Distributed Artificial Intelligence, DAI '25. New York, NY, USA. Association for Computing Machinery, pp. 91–107. <https://doi.org/10.1145/3772429.3772439>.
- Menyhart, O., Weltz, B., Györfy, B., 2021. MultipleTesting.com: a tool for life science researchers for multiple hypothesis testing correction. PLOS One 16, e0245824. <https://doi.org/10.1371/journal.pone.0245824>.
- OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H.W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S.P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S.S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, L., Kamali, A., Kanitscheider, I., Keskar, N.S., Khan, T., Kilpatrick, L., Kim, J.W., Kim, C., Kim, Y., Kirchner, J.H., Kiros, J., Knight, M., Kokotajlo, D., Kondraciuk, L., Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C.M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S.M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mely, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., Peres, F. d. A.B., Petrov, M., Pinto, H.P. d. O., Michael, Pokorny, Pokrass, M., Pong, V.H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F.P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M.B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J.F.C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J.J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C.J., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., Zoph, B., 2024. GPT-4 technical report. <https://doi.org/10.48550/arXiv.2303.08774>.
- Ortega-Ballesteros, A., Manzano-Agugliaro, F., Perea-Moreno, A.-J., 2021. Water utilities challenges: a bibliometric analysis. Sustainability 13, 7726. <https://doi.org/10.3390/su13147726>.
- Qiu, L., Ye, Y., Gao, Z., Zou, X., Chen, J., Gui, Z., Huang, W., Xue, X., Qiu, W., Zhao, K., 2025. Blueprint first, model second: a framework for deterministic LLM workflow. <https://doi.org/10.48550/arXiv.2508.02721>.
- Rossman, L., Woo, H., Tryby, M., Shang, F., Janke, R., Haxton, T., 2020. EPANET 2.2 User Manual. U.S. Environmental Protection Agency, Washington, DC.
- Rossman, L.A., 2022. Storm Water Management Model User's Manual Version 5.2 (No. EPA/600/R-22/083). Office of Research and Development. Center for Environmental & Emergency Response, United States Environmental Protection Agency, Cincinnati, Ohio.
- Sarbu, I., 2021. Optimization of urban water distribution networks using deterministic and heuristic techniques: comprehensive review. J. Pipeline Syst. Eng. Pract. 12, 03121001. [https://doi.org/10.1061/\(ASCE\)PS.1949-1204.0000575](https://doi.org/10.1061/(ASCE)PS.1949-1204.0000575).
- Sarbu, I., Popa-Albu, S., 2023. Optimization of urban water distribution networks using heuristic methods: an overview. Water Int. 48, 120–148. <https://doi.org/10.1080/02508060.2022.2127611>.
- Sela, L., Sowby, R.B., Salomons, E., Housh, M., 2025. Making waves: the potential of generative AI in water utility operations. Water Res. 272, 122935. <https://doi.org/10.1016/j.watres.2024.122935>.
- Shin, W., Souza, R., Rosendo, D., Suter, F., Wang, F., Balaprakash, P., Silva, R.F. d., 2025. The (R)evolution of scientific workflows in the agentic ai era: towards autonomous science. <https://doi.org/10.48550/arXiv.2509.09915>.
- Snell, C., Lee, J., Xu, K., Kumar, A., 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. <https://doi.org/10.48550/arXiv.2408.03314>.
- Taiwo, R., Yussif, A.-M., Zayed, T., 2025. Making waves: generative artificial intelligence in water distribution networks: opportunities and challenges. Water Res. X 28, 100316. <https://doi.org/10.1016/j.wroa.2025.100316>.
- Wang, J., Fu, G., Savic, D., 2026. Leveraging large language models for automating water distribution network optimization. Water Res. 288, 124536. <https://doi.org/10.1016/j.watres.2025.124536>.
- Xu, B., Wu, G., Li, Z., Xu, G., Zeng, H., Tong, R., Ng, H.Y., 2025. Towards domain-adapted large language models for water and wastewater management: methods, datasets and benchmarking. NPJ Clean Water 8, 82. <https://doi.org/10.1038/s41545-025-00509-8>.
- Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., Qiu, Z., 2025. Qwen3 technical report. <https://doi.org/10.48550/arXiv.2505.09388>.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafraan, I., Narasimhan, K., Cao, Y., 2023. ReAct: synergizing reasoning and acting in language models. In: International Conference on Learning Representations. Presented at the International Conference on Learning Representations (ICLR 2023).
- Yu, C., Cheng, Z., Cui, H., Gao, Y., Luo, Z., Wang, Y., Zheng, H., Zhao, Y., 2025. A survey on agent workflow – status and future. In: 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD), pp. 770–781. <https://doi.org/10.1109/ICAIBD64986.2025.11082076>.
- Zanutto, D., Castelletti, A., Savić, D., 2025. Multi-objective water network optimization under shifting reliability perspectives*. In: 2025 6th International Conference on Control and Fault-Tolerant Systems (SysTol). Presented at the 2025 6th International Conference on Control and Fault-Tolerant Systems (SysTol), pp. 331–336. <https://doi.org/10.1109/SysTol66549.2025.11267366>.
- Zhao, Q., Li, D., Liu, Y., Cheng, W., Sun, Y., Oishi, M., Osaki, T., Matsuda, K., Yao, H., Zhao, C., Chen, H., Zhao, X., 2025. Uncertainty propagation on LLM agent. In: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (Eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Presented at the ACL 2025. Association for Computational Linguistics, Vienna, Austria, pp. 6064–6073. <https://doi.org/10.18653/v1/2025.acl-long.302>.
- Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3, 257–271. <https://doi.org/10.1109/4235.797969>.