KGHEDIT: KNOWLEDGE-GRAPH-GUIDED CONTINUAL MODEL EDITING VIA HEAT DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are increasingly deployed in domains with continuously changing knowledge, making post-pretraining updates necessary without repeated full-model retraining or loss of broad model capabilities. Model editing offers a practical path for such deployment-time updates because it can modify parametric knowledge without retraining the full model. However, existing editors remain poorly matched to continual domain updating: they typically treat each update as an independent fact rewrite, even though knowledge is structured and interconnected, and correlated edits can interact through shared entities and relations. We propose **KGHEdit**, a knowledge-graph-guided framework for continual knowledge editing that models incoming edits as local heat sources and uses heat diffusion to identify and weight structurally related support for a base editor. Together, matched controls, interaction-focused streams, and a 10K-edit audit provide convergent evidence that graph-guided support selection and conditional activation offer a practical path to more reliable continual knowledge updating.

1 INTRODUCTION

Large language models are increasingly used in domains where knowledge changes after pretraining. A medical guideline is revised, a financial entity is renamed, or an enterprise record is updated, and the deployed model is expected to absorb the change without losing its general competence Thede et al. (2025); Jin et al. (2021; 2019); Khodja et al. (2024). Continued pretraining can internalize such changes but is often too slow and expensive for frequent updates, even with parameter-efficient variants such as LoRA and QLoRA Hu et al. (2022); Dettmers et al. (2023). Retrieval-augmented generation keeps external knowledge current, but it does not by itself consolidate that knowledge into the model parameters Lewis et al. (2020). Model editing therefore offers a practical path: update parametric knowledge directly at deployment time, without retraining the full model Sinitsin et al. (2020); De Cao et al. (2021); Meng et al. (2022; 2023); Fang et al. (2025).

The central difficulty is not whether a single fact can be rewritten, but whether a model can absorb a *stream* of related updates. Most existing editors are built around a pointwise contract: make the target request true and keep unrelated probes unchanged Meng et al. (2022; 2023); Fang et al. (2025); Mitchell et al. (2022b); Hartvigsen et al. (2023); Wang et al. (2024a). This contract misses the hardest part of continual domain updating, namely the middle region between the edited fact and truly unrelated facts. If a company's headquarters changes, questions about its office address, regional hiring, and local subsidiaries may need to move consistently with the edit, while questions about its founder or product line should remain stable. Treating every non-target fact as protected locality under-propagates the change; writing only the target fact creates inconsistent knowledge; replaying a flat set of neighbors risks overwriting unrelated behavior. The operative question is therefore not simply *what fact should be edited*, but *how far the influence of this edit should travel, and where it should stop*.

This problem becomes sharper when related edits arrive asynchronously. A reliable editor should therefore interpret each request against structurally related current and past updates before changing parameters, so supportive evidence can reinforce the edit while weakly related or operationally colliding facts are screened from the write batch Zhong et al. (2023); Thede et al. (2025); Gupta et al. (2024); Li et al. (2024a).

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.

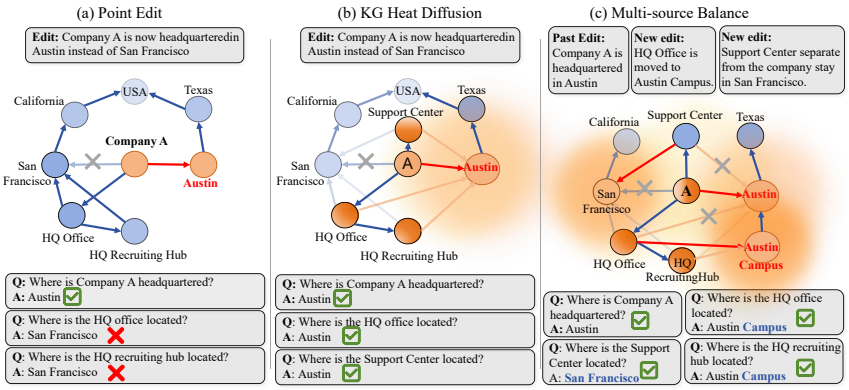


Figure 1: A running example of structure-aware continual editing. (a) A point edit rewrites the target headquarters fact but leaves related queries about the HQ office and recruiting hub inconsistent. (b) KG heat diffusion propagates the edit to nearby related facts while keeping the update local. (c) With historical edits, multiple heat sources are balanced so compatible facts are updated consistently and unrelated facts, such as an independent support center, remain stable.

We argue that continual editing needs an explicit way to estimate edit scope before applying the parameter update. Knowledge graphs are a natural representation for this purpose because they expose entities, relations, and local neighborhoods. Heat diffusion then turns a new edit into graph-side relevance scores: nearby related entities receive higher scores, damping prevents the scores from spreading too far, and linearity lets current and past sources be combined in the same local graph. The base editor still performs the parameter update, but KGHEdit first turns these graph scores into a sparse weighted edit batch that the editor can use.

For each request, KGHEdit links the edit to a local KG neighborhood, solves a history-aware heat-diffusion problem over explicit graph triples, and uses the resulting scores to construct a weighted set of auxiliary facts or prompts for a base editor such as AlphaEdit or NAS Fang et al. (2025); Liu et al. (2026). The scores retrieve current and historical support whose influential nodes overlap the request, after which operational compatibility and locality filters compile a sparse, structured batch for the base editor.

We make the following contributions in this work:

- We formulate continual knowledge editing as a problem of estimating **structured edit scope** rather than repeating independent pointwise fact rewrites, arguing that long-horizon editing fails when correlated updates are treated as unrelated writes.
- We introduce **KGHEdit**, a knowledge-graph-guided framework that uses heat diffusion to score related facts and select weighted auxiliary prompts for existing base editors, enabling related updates to be propagated and balanced before model weights are changed.
- We conduct a multi-scale evaluation across models, editors, domains, and editing horizons of up to 10K updates, demonstrating the practical value of graph-guided scope control for continual knowledge updating.

2 Related Work

Parametric editing. Direct editing methods rewrite model parameters to update factual associations while preserving unrelated behavior. Early methods such as Editable Neural Networks Sinitsin et al. (2020), KnowledgeEditor De Cao et al. (2021), and MEND Mitchell et al. (2022a) learn editing operators from supervision. ROME Meng et al. (2022), MEMIT Meng et al. (2023), and PMET Li et al. (2024b) instead localize factual associations in transformer weights and edit them more explicitly, while AlphaEdit Fang et al. (2025) adds null-space constraints to better preserve prior knowledge under repeated edits. These methods define the strongest in-weight editing baseline for our work, but they still mostly treat each update as an isolated rewrite rather than a structured change over related facts.

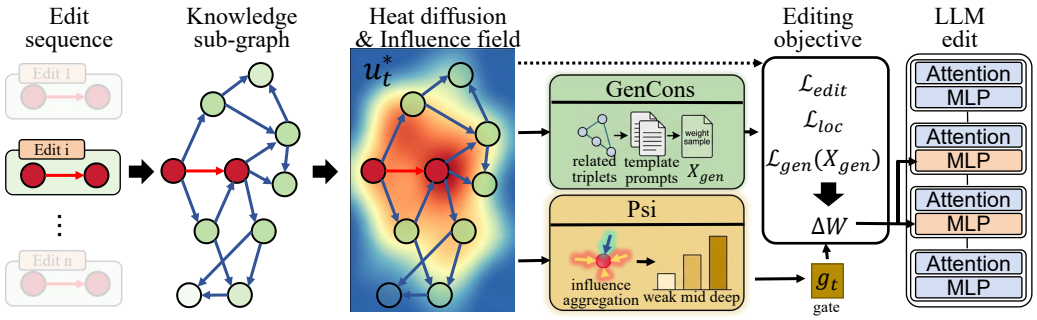


Figure 2: Overview of KGHEdit. Given an edit request, KGHEdit matches the mentioned entities to a local KG subgraph, uses heat diffusion to score related facts, mixes in relevant previous-edit history, selects a weighted support batch, and sends the batch to the base editor.

Continual editing. Once editing is viewed as a long-horizon process rather than a single rewrite, the focus shifts from efficacy to robustness under repeated updates. SERAC Mitchell et al. (2022b) externalizes edits in memory; GRACE Hartvigsen et al. (2023), WISE Wang et al. (2024a), and MELO Yu et al. (2024) reduce interference through key-value memories, revised knowledge memories, or dynamic LoRA-style modules. Recent benchmarks and analyses have correspondingly emphasized multi-hop consistency, temporal updates, and long-horizon forgetting, including MQuAKE Zhong et al. (2023), WikiFactDiff Khodja et al. (2024), WikiBigEdit Thede et al. (2025), and recent evaluation studies on forgetting and protocol mismatch Gupta et al. (2024); Li et al. (2024a). This line is closest to our motivation, but it still mainly handles sequentiality through storage, routing, or adapters. KGHEdit instead asks whether the structural scope of an edit can be inferred before the update is written into model weights.

KG-aware editing. Knowledge graphs explicitly represent entities, typed relations, and local neighborhoods, and GLAME Zhang et al. (2024) uses associated KG knowledge to improve edit generalization. This is the closest prior direction to ours, but its emphasis remains on strengthening individual edits with associated facts. KGHEdit instead treats current and past edits as coupled sources on a local KG neighborhood and uses heat diffusion to select a bounded set of support facts for in-weight editing.

3 Proposed Method: KGHEdit

Continual editing receives requests $\mathcal{R} = \{r_t\}_{t=1}^{T}$, where $r_t = (x_t, y_t)$ asks the model to answer prompt x_t with target y_t. Let θ_{t-1} and θ_t be the editable parameters before and after request t. A base editor changes a small subset of transformer weights, often feed-forward/MLP projection matrices, through rank-one, low-rank, multi-edit, or null-space projected updates Meng et al. (2022; 2023); Fang et al. (2025); Liu et al. (2026). We write this editing step as $\theta_t = \mathcal{E}(\theta_{t-1}, \mathcal{B}_t)$, where $\mathcal{B}_t$ is a weighted list of prompt-target examples (x, y, w). The point-editing baseline uses only $\{(x_t, y_t, 1)\}$, where 1 is unit weight. KGHEdit keeps the same base editor and constructs $\mathcal{B}_t$ before parameter writing; Appendix B gives common weighted-batch losses.

KGHEdit builds $\mathcal{B}_t$ by selecting support examples beyond the target request. Factual edits are not isolated key-value pairs: entities participate in multiple relations, so nearby facts may need to change together while unrelated facts remain stable. The KG first gives a local neighborhood around the request, and heat diffusion turns that neighborhood into relevance scores. The later support-selection step then uses these scores, together with relation metadata and previous-edit history, to decide what should enter $\mathcal{B}_t$, as illustrated in Fig. 2.

3.1 Building a local KG scope

Let $\mathcal{G} = (V, \mathcal{F})$ be a KG with entity set V and typed triples $\mathcal{F}$. For request $r_t = (x_t, y_t)$, KGHEdit first identifies the entities mentioned by the prompt and target, matches them to KG entities, and denotes the matched edit entities by $A_t \subseteq V$. In most factual edits, A_t contains the subject and, when

available, the target object. KGHEdit then expands from A_t to an h-hop local graph $\mathcal{G}_t = (V_t, \mathcal{F}_t)$ under node budget B_g, where $V_t \subseteq V$ and $\mathcal{F}_t \subseteq \mathcal{F}$ are the entities and triples retained for this edit. Index the local entities as $V_t = \{v_i\}_{i=1}^{|V_t|}$. Diffusion runs on a nonnegative undirected multigraph over V_t, where each retained directed KG triple contributes one unit in each direction and parallel relation triples therefore add multiplicity:

$$(W_t)_{ij} = \sum_{(s,r,o)\in\mathcal{F}_t} \left(\mathbf{1}[s = v_i, o = v_j] + \mathbf{1}[s = v_j, o = v_i]\right).$$

Define $D_t = \mathrm{diag}(W_t\mathbf{1})$ and $L_t = D_t - W_t$. This construction preserves relation multiplicity for diffusion and relation labels for support selection; self-loop mass cancels in L_t and contributes zero to the Laplacian quadratic.

3.2 Scoring related facts

The local graph contains many related facts, but not all h-hop facts should enter $\mathcal{B}_t$ with equal weight. KGHEdit uses heat diffusion Zhou et al. (2003); Kondor & Lafferty (2002) to score entities. The current heat-source vector $q_t^{\mathrm{cur}} \in \mathbb{R}^{|V_t|}$ places equal source mass on the edit entities in A_t; Φ_t is the damped diffusion operator, and u_t^{cur} is the resulting entity-score vector:

$$q_t^{\mathrm{cur}}(v) = \begin{cases} 1/|A_t|, & v \in A_t, \\ 0, & v \notin A_t, \end{cases} \qquad \Phi_t = L_t + \mu I, \quad u_t^{\mathrm{cur}} = \Phi_t^{-1} q_t^{\mathrm{cur}}, \quad \mu > 0.$$

KGHEdit solves this damped system and normalizes the resulting heat field for candidate ranking; implementation details are given in Appendix C. This construction ranks entities within the local graph. Heat spreads through KG paths, so nearby related facts receive higher scores; damping keeps the field concentrated near the request. Unlike flat k-hop retrieval, diffusion distinguishes facts within the same neighborhood Zhou et al. (2003); Belkin et al. (2006); Kondor & Lafferty (2002); Gasteiger et al. (2019).

Proposition 1 (Stable diffusion scores)**.** *Let $\mathcal{U}_t = \{\{i, j\} : (W_t)_{ij} > 0,\ i < j\}$ be the undirected edge set used in the diffusion energy. For any heat source $q \in \mathbb{R}^{|V_t|}$ and damping coefficient $\mu > 0$, $\Phi_t u = q$ has the unique solution $u^\star = \Phi_t^{-1} q$, which is also the minimizer of*

$$u^\star = \arg\min_{u\in\mathbb{R}^{|V_t|}} \left[\frac{1}{2}\sum_{\{i,j\}\in\mathcal{U}_t} (W_t)_{ij}(u_i - u_j)^2 + \frac{\mu}{2}\|u\|_2^2 - q^\top u\right]. \tag{1}$$

In this expression, u_i is the heat score of entity i. The edge term smooths scores over KG links, damping keeps the field bounded and local, and $q^\top u$ injects source mass. Proof is given in Appendix E.

3.3 Balancing entity memory

In a stream of edits, a new request may touch a KG neighborhood that already contains earlier edits. Naively reusing past edits ignores KG structure, while independent editing causes forgetting and conflicts Mitchell et al. (2022b); Hartvigsen et al. (2023); Wang et al. (2024a); Gupta et al. (2024); Li et al. (2024a). KGHEdit therefore complements the external KG with a sparse planner-side memory $m_t : V \to \mathbb{R}_{\geq 0}$, initialized to zero, that summarizes the graph scope of preceding requests. Let $\bar{u}_t(v)$ be the normalized heat retained from request t, with zero assigned elsewhere; the memory evolves as

$$m_t(v) = \rho m_{t-1}(v) + \bar{u}_t(v), \qquad \rho \in [0, 1).$$

For a new request, only memory inside the current local graph is read and normalized:

$$a_t(v) = \begin{cases} m_{t-1}(v) \big/ \sum_{z\in V_t} m_{t-1}(z), & v \in V_t,\ \sum_{z\in V_t} m_{t-1}(z) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

With this local readout, a past edit influences the present one only when both meet in the same KG neighborhood. Without the restriction, directly reusing past edits could introduce unrelated facts into the current update.

The current source and local memory source are then mixed as

$$q_t = (1-\alpha)q_t^{\mathrm{cur}} + \alpha a_t, \qquad u_t^\star = \Phi_t^{-1} q_t, \qquad \alpha \in [0,1].$$

The coefficient α controls how much memory influences the current request. The current request and overlapping history act as heat sources on the same graph. Their shared entities receive higher scores, while history outside V_t contributes nothing. The edit-batch step then decides whether an overlap provides support or creates competition Zhou et al. (2003); Kondor & Lafferty (2002); Gasteiger et al. (2019).

Proposition 2 (Structural multi-source reinforcement)**.** *Any local memory source can be written as $a_t = \sum_{j<t} \omega_{j,t} h_{j\to t}$, where $h_{j\to t}$ is the normalized entity-memory footprint of past edit j inside V_t, and $\omega_{j,t} \geq 0$ is its retained weight after decay and local normalization. With $q_t = (1-\alpha)q_t^{\mathrm{cur}} + \alpha a_t$, the mixed heat field $u_t^\star$ is the unique minimizer*

$$u_t^\star = \arg\min_{u \in \mathbb{R}^{|V_t|}} \left[\frac{1}{2} \sum_{\{i,k\} \in \mathcal{U}_t} (W_t)_{ik} (u_i - u_k)^2 + \frac{\mu}{2} \|u\|_2^2 - q_t^\top u \right]. \tag{2}$$

After substituting q_t, the solution becomes $u_t^\star = (1-\alpha)u_t^{\mathrm{cur}} + \alpha \sum_{j<t} \omega_{j,t} u_{j\to t}$, where $u_t^{\mathrm{cur}} = \Phi_t^{-1} q_t^{\mathrm{cur}}$ and $u_{j\to t} = \Phi_t^{-1} h_{j\to t}$. Thus overlapping current and past edits add heat and make shared support candidates easier to select; non-overlapping edits add no local source. Proof is given in Appendix E.

3.4 Constructing the edit batch

The mixed heat field $u_t^\star$ scores candidate relevance; the compiler combines these scores with a same-subject–relation/different-target check to construct the write batch.

Proposition 3 (Attenuation of operational collisions)**.** *Consider two support candidates c_a, c_b that trigger the same-slot/different-target collision screen. A continuous relaxation augments the heat-diffusion energy with a positive balance term of strength γ, encouraging the planner to concentrate heat on one branch. Let $J_\gamma(c_a, c_b)$ denote their joint heat on the shared KG region; its masked-overlap formula is given in Appendix E.4. We can prove that, at the diffusion minimizer,*

$$\frac{d}{d\gamma} J_\gamma(c_a, c_b)\Big|_{\gamma=0} \leq 0.$$

Thus the balance term reduces simultaneous activation to first order and motivates the penalty in the candidate score. Proof is given in Appendix E.

Proposition 3 motivates the selection rule before parameter writing: retain support on high-heat structurally aligned branches and downweight operationally competing branches. KGHEdit turns this principle into a discrete score over a finite candidate set. Current candidates are extracted from local triples in $\mathcal{F}_t$, and memory candidates come from previously submitted planner events whose entity sets overlap V_t. For support candidate c, let $Z(c) \subseteq V_t$ be the entities mentioned by that candidate. KGHEdit ranks c by

$$s(c) = \mathrm{Heat}(c) + \lambda_c \,\mathrm{Compat}(c, r_t) - \lambda_r \,\mathrm{Risk}(c, r_t). \tag{3}$$

Here $\lambda_c, \lambda_r \geq 0$ control the compatibility bonus and conflict-risk penalty. For current candidates, $\mathrm{Heat}(c) = |Z(c)|^{-1} \sum_{v \in Z(c)} u_t^\star(v)$; for memory candidates, it is their heat overlap with the current graph. $\mathrm{Compat}(c, r_t)$ is high when c is likely to reinforce the current edit in the sense of Proposition 2, for example by agreeing with the request relation, prompt template, target type, or edit entities in A_t. $\mathrm{Risk}(c, r_t)$ is high when c is likely to create the kind of competition described in Proposition 3, such as competing for the same subject-relation slot, matching only the relation without target agreement, conflicting in target type, or lying on an overly dense branch. Eq. 3 is the discrete counterpart of the heat-balance principle: heat defines scope, compatibility preserves reinforcing support, and risk suppresses incompatible branches. Appendix C gives the detailed feature definitions and coefficients.

A pre-edit gate activates support when the current request overlaps a previously observed subject–relation slot; otherwise KGHEdit uses the target request alone. When active, KGHEdit keeps up to K

Algorithm 1 KGHEdit for one continual edit

Require: edit request $r_t = (x_t, y_t)$, KG $\mathcal{G}$, model parameters θ_{t-1}, entity memory m_{t-1}
1: Extract edit entities A_t, local graph $\mathcal{G}_t = (V_t, \mathcal{F}_t)$, and operators W_t, L_t, Φ_t
2: Build the current source q_t^{cur} on A_t
3: Restrict entity memory to the local graph and normalize it as a_t
4: $q_t \leftarrow (1-\alpha)q_t^{\mathrm{cur}} + \alpha a_t$; approximately solve $\Phi_t u_t^\star = q_t$
5: Apply the pre-edit gate and rank up to K candidates with Eq. 3
6: Build selected support sets $\mathcal{S}_t^{\mathrm{cur}}, \mathcal{S}_t^{\mathrm{mem}}$ and edit batch $\mathcal{B}_t$
7: Apply the base editor: $\theta_t \leftarrow \mathcal{E}(\theta_{t-1}, \mathcal{B}_t)$
8: Update planner memory m_t with submitted heat scores and decay old memory by ρ
Ensure: edited parameters θ_t

positive-scoring current and memory candidates as $\mathcal{S}_t^{\mathrm{cur}}$ and $\mathcal{S}_t^{\mathrm{mem}}$. Let $\mathcal{S}_t = \mathcal{S}_t^{\mathrm{cur}} \cup \mathcal{S}_t^{\mathrm{mem}}$, and let $[s]_+ = \max(s, 0)$. The weighted interface then normalizes these scores as

$$w_c = \frac{[s(c)]_+}{\sum_{c' \in \mathcal{S}_t} [s(c')]_+}, \qquad \mathcal{B}_t = \{(x_t, y_t, 1)\} \cup \{(\tilde{x}_c, \tilde{y}_c, w_c) : c \in \mathcal{S}_t\},$$

where $(\tilde{x}_c, \tilde{y}_c)$ is the prompt and target of support candidate c. The base editor then writes the transformer parameters from this graph-grounded batch Meng et al. (2023); Fang et al. (2025); Liu et al. (2026).

Algorithm 1 summarizes the steps from graph construction to parameter writing. The remaining question is how the batch $\mathcal{B}_t$ affects the edited model. We analyze one update on target, support, locality, and competing prompts.

Theorem 1 (Conditional local response bound)**.** *Let* $\mathbf{w} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}_t|}$ *collect support weights, let* $\Delta\theta(\mathbf{w}) = \theta_t(\mathbf{w}) - \theta_{t-1}$*, and let* $\eta > 0$ *be the local response coefficient of the base editor. Let* $\bar{\tilde{\Delta}}_g(\mathbf{w})$ *and* $\bar{\Delta}_g(\mathbf{w})$*, for* $g \in \{\mathrm{tar}, \mathrm{com}, \mathrm{loc}, \mathrm{conf}\}$*, denote the first-order and post-edit average target-versus-reference logit-preference changes on the target, structurally aligned, locality, and screened-collision prompt groups, respectively. Eq. 4 summarizes the graph-side plan as a single separation: target and structurally aligned groups should move up, screened-collision groups should move down, and locality should stay close to zero. Define the heat-balance separation*

$$s_t(\mathbf{w}) = \min\Big\{\bar{\tilde{\Delta}}_{\mathrm{tar}}(\mathbf{w}), \bar{\tilde{\Delta}}_{\mathrm{com}}(\mathbf{w}), -\bar{\tilde{\Delta}}_{\mathrm{conf}}(\mathbf{w})\Big\} - \Big|\bar{\tilde{\Delta}}_{\mathrm{loc}}(\mathbf{w})\Big|. \tag{4}$$

Let $s_t^{\mathrm{post}}(\mathbf{w})$ *be the same quantity computed from the post-edit averages* $\bar{\Delta}_g(\mathbf{w})$*. Under the local first-order behavior of the base editor,*

$$s_t^{\mathrm{post}}(\mathbf{w}) \geq \eta\, s_t(\mathbf{w}) - O(\|\Delta\theta(\mathbf{w})\|_2^2). \tag{5}$$

Equation 5 connects a positive first-order batch separation to the realized one-update response up to the editor's second-order remainder. Proof is given in Appendix E.

4 Experiments

Protocol and metrics. We test whether KGHEdit helps a base editor handle long streams of related factual updates. The main benchmark is WikiBigEdit Thede et al. (2025), a sequential editing benchmark built from Wikidata changes; $B100/R20/N2000$ denotes 100 edits per round, 20 rounds, and 2000 total edits, while $N1000$, $N2000$, etc. are shorthand for the total-edit count in the setting column. We also include MedExQA-C, a medical high-conflict stream constructed from MedExQA-style medical QA requests Kim et al. (2024), to test whether graph-guided support remains useful outside WikiBigEdit. CounterFact-10K provides a separate 10,000-edit evaluation with GPT-J-6B and AlphaEdit-FE over three seeds. Each KGHEdit row is matched to its baseline by backbone, editor, request order, stream length, and seed.

The baselines cover direct fine-tuning, lifelong editors, and strong locate-and-edit methods, including FT Wang et al. (2024b), UltraEdit Gu et al. (2026), RLEdit Li et al. (2025), PRUNE Ma et al. (2025),

Table 1: WikiBigEdit results. KGHEdit improves most NAS metrics across three backbones and up to 3,000 edits.

Model	Editor	Setting	Update	Rephrase	Personas	Locality	Prop.
Llama-3.1-8B	FT	B100/R20/N2000	0.1810	0.1088	0.0929	0.0497	0.0886
Llama-3.1-8B	UltraEdit	B100/R20/N2000	0.8178	0.7539	0.6474	0.4575	0.5408
Llama-3.1-8B	RLEdit	B100/R20/N2000	0.4243	0.3964	0.3107	N/A	0.3148
Llama-3.1-8B	PRUNE	B100/R20/N2000	0.2732	0.2710	0.2119	0.2552	0.2028
Llama-3.1-8B	RECT	B100/R20/N2000	0.3208	0.3087	0.2167	0.2722	0.2866
Llama-3.1-8B	AlphaEdit	B100/R20/N2000	0.9541	0.8585	0.7372	0.4456	0.4708
Llama-3.1-8B	AlphaEdit + KGHEdit	B100/R20/N2000	0.9369	**0.8624**	0.7359	0.4309	**0.4930**
Llama-3.1-8B	NAS	B100/R10/N1000	0.9656	0.8595	0.7231	0.4228	0.4409
Llama-3.1-8B	NAS + KGHEdit	B100/R10/N1000	**0.9719**	**0.8781**	**0.7410**	**0.4280**	**0.4614**
Llama-3.1-8B	NAS	B100/R20/N2000	0.8285	0.7147	0.5829	0.4397	0.4826
Llama-3.1-8B	NAS + KGHEdit	B100/R20/N2000	**0.8771**	**0.7513**	**0.6198**	**0.4492**	**0.4835**
Llama-3.1-8B	NAS	B100/R30/N3000	0.8442	0.7329	0.6069	0.4481	0.5163
Llama-3.1-8B	NAS + KGHEdit	B100/R30/N3000	**0.8860**	**0.7649**	**0.6340**	0.4435	**0.5482**
Qwen2.5-7B	NAS	B100/R10/N1000	0.9317	0.7926	0.5105	0.3426	0.2573
Qwen2.5-7B	NAS + KGHEdit	B100/R10/N1000	**0.9726**	**0.8439**	**0.6548**	**0.3495**	**0.3198**
GPT-J-6B	NAS	B100/R10/N1000	0.9891	0.8509	0.6282	0.2602	0.1647
GPT-J-6B	NAS + KGHEdit	B100/R10/N1000	**0.9913**	**0.8622**	**0.6321**	**0.2746**	**0.2114**

Table 2: MedExQA-C results with Llama-3.1-8B. KGHEdit improves all reported metrics.

Editor	Setting	Update	Rephrase	Locality
AlphaEdit	B24/R4/N95	0.9205	0.4075	0.7733
AlphaEdit + KGHEdit	B24/R4/N95	**0.9424**	**0.4292**	**0.8023**
AlphaEdit	B16/R8/N128	0.9283	0.4046	0.7793
AlphaEdit + KGHEdit	B16/R8/N128	**0.9481**	**0.4325**	**0.8132**
AlphaEdit	B16/R10/N154	0.9259	0.3735	0.7703
AlphaEdit + KGHEdit	B16/R10/N154	**0.9475**	**0.3971**	**0.8082**

Table 3: CounterFact-10K results over three seeds. KGHEdit leads the reported policies across editing and preservation metrics.

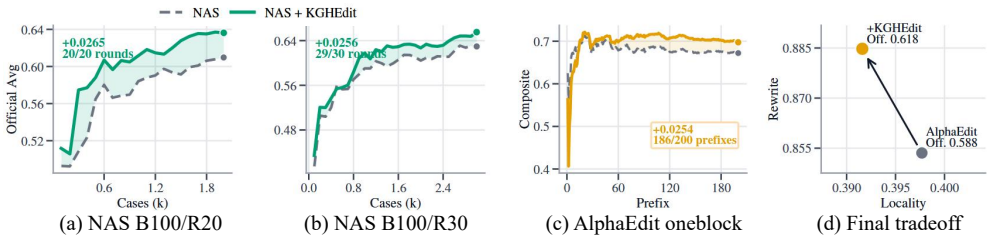
Policy	Target	Rephrase	Locality	Drift ↓	Sq. Drift ↓
Target-only	0.5405	0.4956	0.9813	0.2680	0.3532
Random	0.6653	0.4879	0.9696	0.2785	0.4954
KGHEdit	0.6982	0.5181	**0.9877**	**0.1635**	**0.2420**

RECT Gu et al. (2024), AlphaEdit Fang et al. (2025), and NAS Liu et al. (2026). We report the official axes available for each protocol: **Update** measures the target rewrite, **Rephrase** measures paraphrase generalization, **Personas** measures persona-style prompt robustness, **Locality** measures preservation of protected facts, and **Propagation** measures whether related facts move consistently with the edit. For CounterFact-10K, Target and Rephrase measure editing success, while Drift and Sq. Drift measure changes on protected items. Dataset construction, baseline definitions, metric formulas, hyperparameters, and additional completed analyses are given in Appendices C–D.

Main results. Table 1 reports the expected pattern: KGHEdit is strongest when edits accumulate and begin to interact. With NAS on Llama-3.1-8B, KGHEdit improves every metric at $N1000$ and $N2000$; at $N2000$, Update/Rephrase/Personas rise from 0.8285/0.7147/0.5829 to 0.8771/0.7513/0.6198 while Locality is also higher. At $N3000$, the gains persist on four axes, including Propagation from 0.5163 to 0.5482. This is the regime targeted by KGHEdit: a new edit must be written together with related facts and compatible history, rather than as an isolated point.

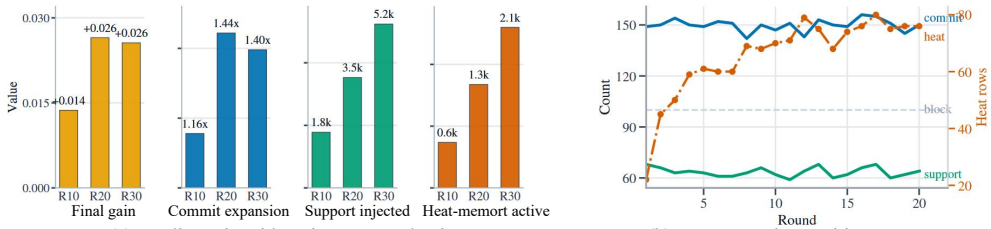
The same pattern is not tied to one base editor or model. With AlphaEdit, KGHEdit improves the two axes most directly tied to structured support, Rephrase and Propagation. With NAS, it improves all five axes on Qwen2.5-7B and GPT-J-6B; the largest cross-model gains appear on Personas and Propagation for Qwen2.5-7B, and on Propagation for GPT-J-6B. On MedExQA-C, all three evaluated axes improve across the $N95$, $N128$, and $N154$ streams. Table 3 extends this advantage to 10,000 edits: KGHEdit leads every reported editing and preservation metric. Its gains over embedding, random, and one-hop selectors (0.0600, 0.0209, and 0.0629) show that graph diffusion selects more useful support than proximity or sampling alone; the 0.0166 gain on later repeats further shows the value of activating that support when edits interact.

Development trajectories. Figure 3 shows that the gains are not confined to final-checkpoint aggregation. KGHEdit outperforms the matched NAS baseline in all 20 rounds of $N2000$ and in 29/30 rounds of $N3000$, so the advantage is maintained throughout the stream rather than appearing



(a) NAS B100/R20 (b) NAS B100/R30 (c) AlphaEdit oneblock (d) Final tradeoff

Figure 3: Trajectory-level evidence for long continual editing. (a–b) Round-wise WikiBigEdit NAS scores for $N2000$ and $N3000$ streams; KGHEdit stays above the matched baseline through almost the entire stream. (c) AlphaEdit oneblock prefix scores. (d) Final rewrite-locality tradeoff, showing that the gain is not limited to target rewriting.



(a) Quality gain with active KG mechanisms (b) Representative positive stream

Figure 4: Mechanism audit on matched runs. (a) Final quality gain together with edit-batch expansion, graph-selected support injection, and heat-memory activations. (b) A representative stream showing that support and memory are used across rounds, matching the intended graph-diffusion editing path.

only after aggregation. This matters for continual editing because early mistakes are repeatedly exposed by later evaluations. With the stronger AlphaEdit oneblock editor, KGHEdit improves 186/200 prefixes and shifts the final rewrite-locality operating point upward. Together, the trajectories and activation traces show that KGHEdit sustains its gains while repeatedly using graph-selected support across the editing stream.

Efficiency comparison. This comparison measures KGHEdit and NAS under the same sequential protocol and reports seconds per edit. Relative cost falls from $1.61\times$ at $N400$ to $1.35\times$–$1.36\times$ on longer streams, showing that graph planning is better amortized as editing continues.

Mechanism audit. Figure 4 shows that the graph-guided support path remains active throughout editing. KGHEdit expands the structured edit batch by $1.156\times$–$1.444\times$, injects 1778–5239 support examples, and activates heat memory 585–2081 times. The representative stream keeps this activity across rounds, which matches the response-field view: the editor should repeatedly receive graph-selected current support and compatible historical support before parameters are written, rather than rely on a single retuned checkpoint.

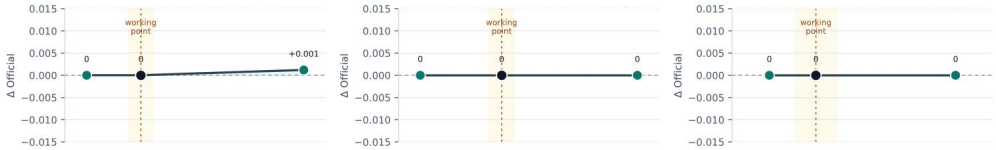
Table 4: Efficiency on matched NAS runs. Relative overhead drops to $1.35\times$–$1.36\times$ on longer streams.

Setting	Method	Sec/Edit	Rel. Time
B20/R20/N400	NAS	7.296	1.00×
B20/R20/N400	NAS + KGHEdit	11.737	1.61×
B100/R20/N2000	NAS	6.764	1.00×
B100/R20/N2000	NAS + KGHEdit	9.203	1.36×
B100/R30/N3000	NAS	6.477	1.00×
B100/R30/N3000	NAS + KGHEdit	8.715	1.35×

Ablation. Table 5 shows that diffusion-based ranking and multi-source balance provide complementary gains. Replacing heat diffusion with entity retrieval or removing heat from the support path lowers all target-side metrics, showing that KGHEdit needs smooth heat-based ranking rather than a flat neighbor list. The no-balance row is especially diagnostic: target rewriting remains strong, but Propagation drops from 0.2792 to 0.2375. This is the expected role of multi-source balance: it turns overlapping current and historical heat into coherent graph-level updates while suppressing incompatible branches.

Table 5: Component ablation on the strict proxy stream. All rows use the same base editor, request order, support budget, repetition cap, and evaluator; only the scope rule and balance switch change. Full KGHEdit strengthens target-side editing over retrieval, PPR, and no-heat variants, while multi-source balance improves structured propagation.

Scope rule	KG graph	Heat eq.	Balance	Update	Rephrase	Personas	Locality	Prop.
Heat equilibrium (ours)	✓	✓	✓	**0.7836**	**0.6892**	**0.5346**	**0.4391**	**0.2792**
Discrete k-hop	✓	×	✓	0.7662	0.6699	0.5136	0.4166	0.2644
Entity retrieval	×	×	✓	0.7519	0.6543	0.5040	0.4160	0.2642
Personalized PageRank	✓	×	✓	0.7587	0.6584	0.5025	0.4216	0.2662
Support path, no heat	✓	×	✓	0.7508	0.6553	0.5027	0.4169	0.2664
Heat eq., no balance	✓	✓	×	0.7740	0.6821	0.5209	0.4197	0.2375



(a) Heat-memory mixing α (b) Pair-compatibility gate τ_{pair} (c) Replay gate τ_{xmem}

Figure 5: Stress-flow robustness on a mixed stream with related, unrelated, and conflicting edits. The three panels vary heat-memory mixing α, pair-compatibility threshold, and memory-support gate under low/default/high settings; KGHEdit remains on a stable performance plateau.

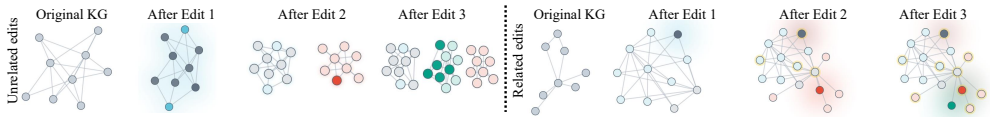


Figure 6: Three-edit KG growth case study. Unrelated edits remain in separated heat regions, while related edits share compatible heat and conflicting branches are split before parameter writing. Halos denote shared heat; separated branches denote unrelated or conflict-suppressed scope.

Development sensitivity and cases. Figure 5 varies the compatibility threshold, heat-memory weight, and memory replay gate on a mixed stream containing related, conflicting, and unrelated edits. On this single development flow, low/default/high settings have similar final scores; this is a descriptive scan rather than a broad robustness guarantee. Figure 6 provides a wide three-edit trace of the corresponding scope decisions: unrelated edits remain in separate heat regions, compatible plant-range edits share the $P9714 \rightarrow$ Fujian structure, and conflicting object branches are suppressed before parameter writing. Appendix D gives the concrete traces.

5 Conclusion and Limitation

We presented KGHEdit, a structure-aware continual editing framework that maps each request to a heat-scored KG neighborhood and turns selected facts into a sparse support batch for a base editor. Heat diffusion proposes relevant scope, while the support compiler screens operational collisions and the pre-edit gate activates support only in an observable interaction regime. This makes KGHEdit a lightweight graph-side layer for AlphaEdit, NAS, or other batched editors. Equal-budget controls show gains over embedding, random, and one-hop selection; the positive later-repeat gate test and complete 10K target–preservation audit further support conditional graph-guided editing. Its graph-side support depends on entity linking and KG coverage, and future work can learn or refresh that support for noisy, private-domain, or rapidly changing knowledge.

AI Use Statement

Generative AI tools were used solely for language editing. The authors verified the final manuscript and take full responsibility for the submission.

Ethics Statement

This work evaluates continual knowledge editing in large language models using existing benchmarks and constructed evaluation streams, without involving human subjects or collecting personal data. Knowledge editing can support timely factual updates, but may also be misused to introduce or propagate false information, particularly in high-stakes domains such as medicine. Deployment should therefore verify the provenance and correctness of edits and maintain appropriate safeguards and auditing mechanisms.

Reproducibility Statement

The supplementary artifact provides machine-readable numerical summaries and maps every reported table and figure to its source measurements. The appendix specifies environments, task contexts, training schedules, evaluation units, statistical procedures, and complete proofs.

References

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7: 2399–2434, 2006. URL https://jmlr.org/papers/v7/belkin06a.html.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.522. URL https://aclanthology.org/2021.emnlp-main.522/.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, pp. 10088–10115. Neural Information Processing Systems Foundation, 2023. doi: 10.52202/075280-0441. URL https://arxiv.org/abs/2305.14314.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.02355. URL https://arxiv.org/abs/2410.02355.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, pp. 13333–13345, 2019. URL https://arxiv.org/abs/1911.05485.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. doi: 10.48550/arXiv.2401.04700. URL https://arxiv.org/abs/2401.04700.

Xiaojie Gu, Ziying Huang, Jia-Chen Gu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in language models. *Transactions on Machine Learning Research*, 2026. doi: 10.48550/arXiv.2505.14679. URL https://arxiv.org/abs/2505.14679.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15202–15232. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.902. URL https://aclanthology.org/2024.findings-acl.902/.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*, pp. 47934–47959. Neural Information Processing Systems Foundation, 2023. doi: 10.52202/075280-2079. URL https://arxiv.org/abs/2211.11031.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2106.09685. URL https://arxiv.org/abs/2106.09685.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421. URL https://arxiv.org/abs/2009.13081.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2567–2577. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259/.

Hichem Ammar Khodja, Frédéric Béchet, Quentin Brabant, Alexis Nasr, and Gwénolé Lecorvé. WikiFactDiff: A large, realistic, and temporally adaptable dataset for atomic factual knowledge update in causal language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 17614–17624. ELRA and ICCL, 2024. doi: 10.63317/5kmnn4c4u35s. URL https://arxiv.org/abs/2403.14364.

Yunsoo Kim, Jinge Wu, Yusuf Abdulle, and Honghan Wu. MedExQA: Medical question answering benchmark with multiple explanations. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, 2024. doi: 10.48550/arXiv.2406.06331. URL https://arxiv.org/abs/2406.06331.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning*, pp. 315–322, 2002. URL https://dl.acm.org/doi/10.5555/645531.656023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, pp. 9459–9474, 2020. URL https://arxiv.org/abs/2005.11401.

Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *Advances in Neural Information Processing Systems*, pp. 30850–30885. Neural Information Processing Systems Foundation, 2024a. doi: 10.52202/079017-0971. URL https://openreview.net/forum?id=m0DS4OOmSY.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18564–18572, 2024b. doi: 10.1609/AAAI.V38I17.29818. URL https://dblp.org/rec/conf/aaai/Li0SYMY24.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *International Conference on Machine Learning*, 2025. doi: 10.48550/arXiv.2502.05759. URL https://arxiv.org/abs/2502.05759.

Mingda Liu, Zhenghan Zhu, Ze'an Miao, and Katsuki Fujisawa. Toward ultra-long-horizon sequential model editing. *arXiv preprint arXiv:2602.02543*, 2026. URL https://arxiv.org/abs/2602.02543.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2405.16821. URL https://arxiv.org/abs/2405.16821.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, pp. 17359–17372. Neural Information Processing Systems Foundation, 2022. doi: 10.52202/068431-1262. URL https://arxiv.org/abs/2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2210.07229. URL https://arxiv.org/abs/2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. doi: 10.48550/arXiv.2110.11309. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 15817–15831, 2022b. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. In *International Conference on Learning Representations*, 2020. doi: 10.48550/arXiv.2004.00345. URL https://arxiv.org/abs/2004.00345.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Tom Hartvigsen. Wikibigedit: Understanding the limits of lifelong knowledge editing in llms. *arXiv preprint arXiv:2503.05683*, 2025.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems*, pp. 53764–53797. Neural Information Processing Systems Foundation, 2024a. doi: 10.52202/079017-1703. URL https://arxiv.org/abs/2405.14768.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 82–93. Association for Computational Linguistics, 2024b. doi: 10.48550/arXiv.2308.07269. URL https://aclanthology.org/2024.acl-demos.9/.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19449–19457, 2024. doi: 10.1609/AAAI.V38I17.29916. URL https://dblp.org/rec/conf/aaai/YuCZH24.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22647–22662. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1261. URL https://aclanthology.org/2024.emnlp-main.1261/.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.971. URL https://arxiv.org/abs/2305.14795.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2003. URL https://papers.nips.cc/paper/2506-learning-with-local-and-global-consistency.

Table 6: Main notation used by KGHEdit.

Group	Symbol	Meaning
Editing interface		
Edit stream	$\mathcal{R} = \{r_t\}_{t=1}^{T}$	Sequence of edit requests.
Request	$r_t = (x_t, y_t)$	Prompt x_t and target answer y_t at edit order t.
Editable parameters	θ_{t-1}, θ_t	Parameters changed by the base editor before and after edit t.
Base editor	$\mathcal{E}$	Underlying editor such as AlphaEdit or NAS.
Edit batch	$\mathcal{B}_t$	Target edit plus selected support examples sent to the base editor.
Support sets	$\mathcal{S}_t^{\mathrm{cur}}, \mathcal{S}_t^{\mathrm{mem}}$	Current-KG and memory support selected by KGHEdit.
Support item	$(\tilde{x}_i, \tilde{y}_i, w_i)$	Prompt, target, and weight for support example i.
KG scope and heat diffusion		
Knowledge graph	$\mathcal{G} = (V, \mathcal{F})$	Entity set V and factual triples $\mathcal{F}$.
Linked edit entities	A_t	Entities linked from the edit request, usually subject/object entities.
Local graph	$\mathcal{G}_t = (V_t, \mathcal{F}_t)$	h-hop KG neighborhood under node budget B_g.
Adjacency	W_t	Weighted local adjacency matrix after symmetrization.
Undirected edge set	$\mathcal{U}_t$	Edges induced by nonzero entries of W_t, used in the heat energy.
Laplacian system	D_t, L_t, Φ_t	D_t is the degree matrix, $L_t = D_t - W_t$, and $\Phi_t = L_t + \mu I$.
Current source	q_t^{cur}	Normalized source vector over the matched edit entities in V_t.
Current heat	u_t^{cur}	Solution of $\Phi_t u_t^{\mathrm{cur}} = q_t^{\mathrm{cur}}$.
Mixed heat	$u_t^{\star}$	Final heat score after current-memory source mixing.
Edit memory and support selection		
Entity memory	$m_t : V \to \mathbb{R}_{\geq 0}$	Sparse global memory over entities after edit t.
Decay and mixing	ρ, α	Memory decay and current-memory source mixing coefficients.
Local memory source	a_t	m_{t-1} restricted to the current local graph V_t and normalized.
Restricted past source	$h_{j\to t}, \omega_{j,t}$	Source of past edit j on V_t and its retained decayed weight.
Mixed source	q_t	$(1-\alpha)q_t^{\mathrm{cur}} + \alpha a_t$, the source used in $\Phi_t u_t^{\star} = q_t$.
Candidate scores	$\mathrm{Heat}, \mathrm{Compat}, \mathrm{Risk}$	Heat support, semantic compatibility, and conflict-risk terms used in $s(c)$.
Score weights	λ_c, λ_r	Compatibility and conflict-risk weights in Eq. 3.
Local first-order analysis		
Grouped responses	$\bar{\tilde{\Delta}}_g, \bar{\Delta}_g$	First-order and post-edit average logit-preference changes for $g \in \{\mathrm{tar}, \mathrm{com}, \mathrm{loc}, \mathrm{conf}\}$.
Heat-balance separation	$s_t(\mathbf{w})$	First-order separation between target/compatible groups and competing/locality groups in Theorem 1.
Conflict set	$\mathcal{C}_{ab}, \lambda_r$	Conflict entities and penalty strength for incompatible source overlap.

A Notation Summary

B Editing Losses for Base Editors

The base-editor interface in Section 3 can be implemented by different editing objectives. Most parametric editing methods combine three ingredients: an example loss that makes the requested answer likely, a locality term that keeps unrelated behavior close to the pre-edit model, and editor-specific constraints that determine how the update is written into parameters De Cao et al. (2021); Mitchell et al. (2022a); Meng et al. (2022; 2023); Fang et al. (2025); Wang et al. (2024a). Let $p_\theta(\cdot \mid x)$ denote the language model distribution, and let $y = (y_1, \ldots, y_m)$ be the target token sequence.

Target edit loss. The standard implementation is the teacher-forced negative log-likelihood of the new answer:

$$\mathcal{L}_{\mathrm{edit}}(\vartheta; r_t) = -\sum_{\ell=1}^{m} \log p_\vartheta(y_{t,\ell} \mid x_t, y_{t,<\ell}) \,. \tag{6}$$

For training-style editors such as KnowledgeEditor and MEND, this loss directly trains the editing network or the edited parameters De Cao et al. (2021); Mitchell et al. (2022a). For locate-and-edit methods such as ROME, MEMIT, and AlphaEdit, the same likelihood objective appears inside the value-optimization step: the method searches for a layer value or weight update that makes prompts about the edited subject predict the new object Meng et al. (2022; 2023); Fang et al. (2025). This loss implements the minimum requirement of editing: after the update, the model should assign high probability to the desired answer for the target request.

Locality loss. Editing should not change prompts that are unrelated to the requested fact. A common way to express this requirement is a distribution-matching penalty on protected prompts $\mathcal{X}_t^{\mathrm{loc}}$:

$$\mathcal{L}_{\mathrm{loc}}(\vartheta, \theta_{t-1}) = \frac{1}{|\mathcal{X}_t^{\mathrm{loc}}|} \sum_{x \in \mathcal{X}_t^{\mathrm{loc}}} \mathrm{KL}\big(p_{\theta_{t-1}}(\cdot \mid x) \,\big\|\, p_{\vartheta}(\cdot \mid x)\big)\,. \tag{7}$$

Some implementations compute the KL divergence over the next-token distribution; others use answer-level likelihood or exact-match probes during evaluation. The purpose is the same: the pre-edit model $p_{\theta_{t-1}}$ acts as a teacher on protected prompts, so the edited model is discouraged from moving probability mass on facts that should remain unchanged. This is the locality or preservation term used, explicitly or implicitly, by many editing systems and by benchmark locality probes Sinitsin et al. (2020); De Cao et al. (2021); Meng et al. (2022; 2023); Fang et al. (2025); Thede et al. (2025).

Weighted support examples. KGHEdit extends the editor input by adding graph-supported current facts and compatible memory examples to $\mathcal{B}_t$. If the selected support items are $\mathcal{S}_t = \{(\tilde{x}_i, \tilde{y}_i, w_i)\}_{i=1}^{N_t}$, their contribution to the weighted example loss is

$$-\sum_{i=1}^{N_t} w_i \sum_{\ell=1}^{|\tilde{y}_i|} \log p_{\vartheta}(\tilde{y}_{i,\ell} \mid \tilde{x}_i, \tilde{y}_{i,<\ell})\,, \qquad \sum_i w_i = 1, \quad w_i \geq 0. \tag{8}$$

If a base editor does not expose weighted examples, KGHEdit implements the same preference by repeating high-score examples up to a small cap. Current support makes related facts move with the target edit, while memory support preserves compatible earlier edits that overlap the current KG neighborhood. These examples are therefore not generic data augmentation: their selection and weights are determined by heat scores, compatibility, and conflict risk in Eq. 3 before the base editor writes parameters.

Support selection features. Table 7 instantiates the quantities in Eq. 3. The support selector keeps the graph-derived terms separate from relation and type checks: heat selects where support should come from, compatibility raises structurally matched candidates, and the risk term suppresses candidates that would compete with the current edit before the base editor writes parameters. For memory pairs, the compatibility score is computed as

$$\begin{aligned} C = {} & 0.13c_{\mathrm{rel}} + 0.08c_{\mathrm{fam}} + 0.07c_{\mathrm{temp}} + 0.10c_{\mathrm{role}} + 0.14o_{\mathrm{clus}} + 0.10o_{\mathrm{tri}} \\ & + 0.10c_{\mathrm{type}} + 0.08c_{\mathrm{targ}} + 0.10o_{\text{targ-anc}} + 0.15o_{\text{targ-clus}} + 0.04o_{\mathrm{hot}} + 0.02o_{\mathrm{seed}}, \end{aligned}$$

where c terms are binary relation, template, role, and target-type matches, and o terms are normalized overlaps over the heat neighborhood, KG triples, and target anchors. The development runs use a pair-compatibility working point of 0.30, an absolute support cap of 40, and bounded support repetition with cap 2.

Table 7: Deterministic features used by support selection.

Term	Source	Implementation	Effect in support selection
$\mathrm{Heat}(c)$	Current or memory heat	Average heat on the entities $Z(c)$ of a selected current or memory candidate.	Ranks support by edit-induced graph heat.
$\mathrm{Compat}(c, r_t)$	Relation/type metadata	Weighted match over relation id, relation family or role, prompt template, target type/signature, and hot-node, seed, cluster, triple, and target-anchor overlap.	Promotes candidates that share the same structural role as the current edit.
$\mathrm{Risk}(c, r_t)$	Slot and support checks	Penalty from competing subject–relation slots, relation-only matches without target support, incompatible target type, oversized local branches, or low net support score.	Lowers candidates likely to over-propagate or reinforce an incompatible branch.
Caps	Support budget	Per-edit current and memory caps, per-relation cap, absolute support cap, and bounded repetition for unweighted editors.	Keeps the selected batch sparse and matched across ablations.

C Experimental Details

Dataset. The original long-stream and MedExQA analyses below are retained as development diagnostics; confirmatory claims use the outcome-blind matrix and separately audited results in the

main text. WikiBigEdit is a large-scale continual editing benchmark derived from real Wikidata changes Thede et al. (2025). Its sequential editing format presents edit requests in a fixed order and groups them into batches. An edit item contains a target prompt and answer, plus auxiliary probes used by the official evaluator: a rephrased prompt for generalization, persona-style prompts for robustness under surface-form changes, protected locality prompts for nearby facts that should remain stable, and propagation prompts for related facts that should move consistently with the update. The notation $B/R/N$ records the batch size, number of rounds, and total number of requested edits; for instance, $B100/R20/N2000$ edits 100 requests per round for 20 rounds. The main WikiBigEdit stream uses Llama-3.1-8B and preserves the benchmark request order. The Qwen2.5-7B and GPT-J-6B rows apply the same WikiBigEdit protocol to additional backbones and show that KGHEdit transfers useful graph-guided support beyond the main Llama setting. MedExQA-C contains medical high-conflict streams constructed from MedExQA-style medical QA requests Kim et al. (2024) with paired or competing subjects. We report three matched variants: a conflict-priority stream, a support-mix stream, and a longer support-heavy stream. These streams test whether graph-guided support selection remains useful beyond the original WikiBigEdit distribution.

MedExQA-C construction. MedExQA-C is a constructed evaluation stream for the medical transfer check. We start from MedEditBench MedExQA edit requests and retain medical QA items with normalized subject, answer, relation-family, and template signatures. Requests are grouped by relation family, require answer diversity, and are arranged into mixed streams that include natural same-subject bundles and paired competing-answer bundles. The conflict-priority setting contains 95 edits over 41 subjects; the support-mix setting contains 128 edits over 56 subjects; the support-heavy setting contains 154 edits over 66 subjects. All three variants use fixed request files, matched seed0/seed1 pairs, and the same strict domain-update evaluator. The edit-time support selector receives only target request fields and graph-derived support fields; rephrase and locality probes are reserved for evaluation.

Table 8: Information boundary for the editing protocols.

Field	Edit-time support selector	Evaluator	Role
Target prompt and target answer	✓	✓	Requested factual rewrite.
Subject, object, relation id, template signature	✓	×	KG anchoring, relation-family checks, and support construction.
Local KG triples and heat-memory records	✓	×	Candidate support, heat diffusion, and compatible memory replay.
Support-source tags and selection scores	✓	×	Batch weighting, caps, and bounded repetition.
Rephrase prompts and answers	×	✓	Generalization evaluation.
Persona prompts and answers	×	✓	Surface-form robustness evaluation when available.
Locality prompts and protected answers	×	✓	Preservation evaluation.
Propagation prompts and related answers	×	✓	Related-fact consistency evaluation when available.

Baselines. We group baselines by their editing mechanism and evaluate all available outputs with the same final-stream evaluator. **FT**[1] is a training-style lower-bound baseline implemented with EasyEdit Wang et al. (2024b): for each batch, it optimizes the edited examples directly with supervised loss and therefore tests whether ordinary fine-tuning can survive the same sequential protocol. **UltraEdit**[2] is a lifelong editing baseline that computes lightweight parameter shifts without subject lookup or external memory and maintains running normalization statistics for scalable editing Gu et al. (2026). **RLEdit**[3] formulates lifelong editing as reinforcement learning over hypernetwork-generated updates, using sequence-level editing rewards to adapt the editor to changing model parameters Li et al. (2025). **PRUNE**[4] is a locate-and-edit stabilizer that constrains the condition number of edited matrices so that repeated edits perturb stored knowledge less aggressively Ma et al. (2025). **RECT**[5] regularizes edit updates by their relative weight change, aiming to reduce damage to general abilities while

[1] https://github.com/zjunlp/EasyEdit
[2] https://github.com/XiaojieGu/UltraEdit
[3] https://github.com/zhrli324/RLEdit
[4] https://github.com/mjy1111/PRUNE
[5] https://github.com/JasonForJoy/Model-Editing-Hurt

retaining the principal editing signal Gu et al. (2024). **AlphaEdit**[6] is a strong locate-and-edit method that projects the edit perturbation into the null space of preserved knowledge Fang et al. (2025). **NAS**[7] is Norm-Anchor Scaling, a plug-in stabilizer that rescales locate-and-edit updates to control long-horizon weight-norm growth Liu et al. (2026). KGHEdit does not replace any of these base editors. In matched KGHEdit runs, it anchors each request to the KG, computes heat-guided scores, filters compatible support and memory examples, and passes the resulting structured edit batch to the same base editor before the editor writes parameters.

Metrics. We follow the WikiBigEdit final-stream evaluation axes. For each edit item i, let $\mathcal{P}_i^{\mathrm{upd}}$, $\mathcal{P}_i^{\mathrm{rep}}$, $\mathcal{P}_i^{\mathrm{per}}$, $\mathcal{P}_i^{\mathrm{loc}}$, and $\mathcal{P}_i^{\mathrm{prop}}$ denote, respectively, target update prompts, rephrased prompts, persona-style prompts, protected locality prompts, and propagation prompts. Each probe is a pair (x, y), where y is the expected answer under that axis: the new answer for update/rephrase/persona probes, the protected old answer for locality probes, and the expected changed related answer for propagation probes. Let $\hat{y}_{\theta_T}(x)$ be the answer produced by the model after the final editing round, and let $c(\hat{y}, y) \in \{0, 1\}$ be the official normalized exact-match predicate. For an axis a, the score is

$$S_a = \frac{\sum_i \sum_{(x,y)\in\mathcal{P}_i^a} c(\hat{y}_{\theta_T}(x), y)}{\sum_i |\mathcal{P}_i^a|}. \tag{9}$$

Thus S_{upd} measures target rewrite success, S_{rep} paraphrase generalization, S_{per} persona robustness, S_{loc} preservation of protected facts, and S_{prop} consistency on related facts that should move with the edit. WikiBigEdit evaluates all five axes. The MedExQA-C protocol evaluates Update, Rephrase, and Locality, but does not define persona or propagation probes; those cells are marked as "–" and are excluded from comparison. Round-wise curves use the same formula with θ_T replaced by the checkpoint after the corresponding round and the prefix evaluated by that round.

Setup. The development diagnostics use the strict WikiBigEdit multiblock protocol with final-stream evaluation; the outcome-blind N=320 control instead fixes active $K = 5$ and every model-facing weight to one. Matched pairs share the same model, request order, block size, number of rounds, and random seed. For Llama-3.1-8B AlphaEdit and NAS, we edit layers 4–8 of the MLP down projection, use subject-last fact tokens, 25 value-optimization steps, $v_{\mathrm{lr}} = 0.1$, $\lambda_{\mathrm{KL}} = 0.0625$, second-moment statistics from Wikipedia with 32 samples, and AlphaEdit null-space threshold 2×10^{-2}. NAS additionally uses online decay 0.95, skip factor 2.0, clip factor 2.0, and a 64-edit warm-up buffer. KGHEdit uses uniform edge weights on explicit KG triples, damping $\mu = 0.1$, one-hop task neighborhoods, the top 20 heat-ranked task nodes, heat-memory weight $\alpha = 0.2$, memory decay 0.98, and at most 50k retained heat-memory entries. For support construction, we select up to 20 support facts per edit with coupled current-memory budgeting: the memory-support ratio is 0.30, the absolute support cap is 40, the minimum support score is 0.20, the hot-node override threshold is 0.45, the per-relation cap is 3, the repeated-example cap is 2 with step 0.25, and the pair-compatibility threshold is 0.30. The MedExQA-C rows use the strict domain-update protocol with $B24/R4/N95$, $B16/R8/N128$, and $B16/R10/N154$; each row averages seed0/seed1 matched pairs. Efficiency numbers are measured end-to-end per requested edit, including graph anchoring, diffusion, support selection, base-editor execution, and evaluation.

C.1 Implementation details

KGHEdit uses one fixed support-selection working point for the reported WikiBigEdit and MedExQA-C runs unless an ablation explicitly changes a single component. The $\mathrm{Compat}/\mathrm{Risk}$ coefficients in Eq. 3 and Table 7 are the implementation coefficients used by support selection; they are held fixed across model backbones, seeds, and stream lengths. The default pair-compatibility gate 0.30, heat-memory mixing $\alpha = 0.20$, and memory-support gate 0.10 are the middle values in the stress-flow scan in Table 12, where the low/default/high settings remain on the same performance plateau.

The reported runs are specified by protocol-level fields: model backbone, base editor, $B/R/N$, request order, seed/order setting, evaluator axes, and the fixed KGHEdit support-selection working point. Matched baseline/KGHEdit pairs share these fields and differ only in whether graph-side

[6] https://github.com/jianghoucheng/AlphaEdit
[7] https://github.com/SasyaTitech/NAS

support examples are supplied before the base editor writes parameters. WikiBigEdit uses final-stream evaluation on the official Update, Rephrase, Personas, Locality, and Propagation axes; MedExQA-C uses the domain-update axes reported in Table 1. For support construction, the fixed working point uses `plugin_top_nodes=20`, `support_k=20`, memory-support ratio 0.30, absolute support cap 40, budget cap 60, per-relation cap 3, per-edit current/memory caps $1/1$, pair gate 0.30, and bounded repetition cap 2.

The intervals in Table 10 are paired trajectory diagnostics: each delta compares KGHEdit and its base editor after the same round, with the same edited prefix and the same request order. The reported intervals summarize within-trajectory evolution. Seed/order fields are part of the protocol record, and each reported baseline/KGHEdit comparison keeps these fields matched, isolating graph-side support selection under the same request sequence.

C.2 KG COVERAGE

The edit-time support selector records request-schema diagnostics before model loading. Table 9 reports coverage checks used by the WikiBigEdit and MedExQA-C support streams: the sampled streams have complete relation-id fields, a broad set of relation anchors, and repeated subjects that make compatible-memory reuse observable. These diagnostics are computed from edit-time request metadata; evaluator-only fields remain outside the support-selection boundary defined in Table 8.

Table 9: KG coverage and anchor diagnostics for representative edit streams. Repeated subjects count subjects appearing at least twice in the sampled stream.

Stream	N	Seed/order	Missing relation id	Unique relation ids / repeated subjects
WBE B100/R10/N1000	1000	seed0	0	162 / 10
WBE B100/R20/N2000	2000	seed0	0	213 / 43
WBE B100/R30/N3000	3000	seed0	0	244 / 96
WBE B100/R20/N2000	2000	seed1	0	197 / 40
MedExQA-C support-mix	128	fixed seed pair	0	72 / 56
MedExQA-C support-heavy	154	fixed seed pair	0	84 / 66

When the local KG around an edit is sparse, KGHEdit keeps the target edit and its canonical anchor in the selected batch, then adds only support examples that pass the heat, compatibility, conflict-risk, and cap checks. Thus sparse neighborhoods reduce the support set instead of triggering unrelated replay. The case traces in Appendix D show the same support-selection behavior at the level of concrete edit histories: compatible branches reuse shared anchors, unrelated branches stay separated, and competing target branches are filtered before parameter writing.

D ADDITIONAL EXPERIMENTAL ANALYSIS

Mechanism audit. Table 11 reports the detailed counters behind Figure 4. The score gains in Table 1 occur together with larger structured edit batches, nonzero support injection, and repeated heat-memory activation. These traces show that KGHEdit contributes through the intended heat-based support path before the base editor writes parameters.

Table 10: Paired trajectory intervals for the main WikiBigEdit NAS rows. The interval is computed over round-wise KGHEdit–NAS official-score deltas on the same matched stream.

Setting	Mean delta	95% interval
NAS B100/R10/N1000	$+0.0145$	$[0.0067, 0.0223]$
NAS B100/R20/N2000	$+0.0310$	$[0.0253, 0.0366]$
NAS B100/R30/N3000	$+0.0196$	$[0.0166, 0.0227]$

Table 11: Detailed mechanism counters corresponding to Figure 4. Edit-batch expansion is relative to the baseline number of requested edits.

Run	Edit-batch expansion	Support injected	Heat-memory active	Local KG size
NAS B100/R10/N1000	1.156×	1778	585	245.7 nodes / 337.2 triples
NAS B100/R20/N2000	1.444×	3526	1343	241.0 nodes / 331.3 triples
NAS B100/R30/N3000	1.396×	5239	2081	241.9 nodes / 334.8 triples

Case-study details. Appendix Figures 7–10 expand the case study in Figure 6. The examples are selected to make the scope decision explicit: KGHEdit should keep unrelated histories apart, reuse support when edits share compatible KG structure, and suppress overlap when it comes from incompatible branches.

Unrelated edits. In Figure 7, the two-step case edits CFGT-FM → Cogeco Media under $P127$ and *Bambusa beecheyana* → Fujian under $P9714$. The first edit enters a media-ownership neighborhood, while the second enters a plant-geography neighborhood. The cumulative KG therefore grows, but the heat regions stay on different components. This is the behavior KGHEdit is designed to produce: edit order alone does not make two facts share support, so an ownership update does not inject plant-range evidence into the base editor.

The three-step unrelated case adds Pierre Wibaux House → NRHP building under $P2868$. This creates a third heritage-designation component, and the blue, orange, and green heat regions attach to different local structures. This case illustrates locality at the graph level rather than only at the prompt level. KGHEdit can accept a longer local history while still compiling separate support sets for unrelated facts.

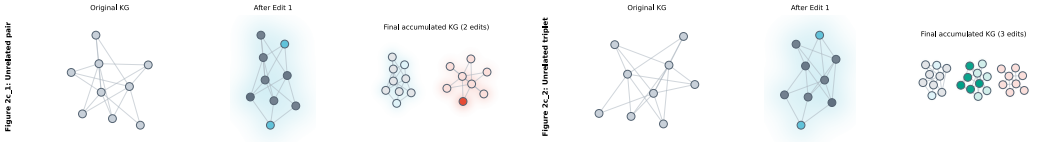


Figure 7: Unrelated-edit cases. Left: a two-step ownership/range pair stays split into two components. Right: a third heritage-designation edit creates a third component rather than merging histories.

Compatible edits. Figure 8 shows the opposite decision. In the geography pair, two plant-range edits point to the same Fujian target under $P9714$. The second edit does not start a disconnected component; it expands the existing Fujian-centered skeleton. KGHEdit treats this overlap as useful evidence, so support selected for the second edit can reuse the region structure created by the first edit.

The geography triplet strengthens the same point with *Bambusa beecheyana*, *Parathelypteris angustifrons*, and *Smilax bracteata*, all writing $P9714$ → Fujian. Even when the third edit adds few new nodes, its heat field overlaps the existing Fujian skeleton. This is the intended role of multi-source heat: compatible updates reinforce a stable scope, allowing the base editor to update a range-level structure consistently without expanding to unrelated entities.

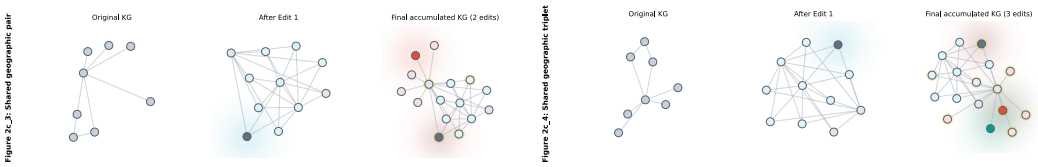


Figure 8: Compatible geography cases. Different plant subjects point to Fujian, so later edits grow the same skeleton instead of starting isolated components.

Figure 9 includes a second compatible example from a different relation family. Two radio-station ownership edits share Cogeco Media under $P127$. The cumulative graph grows along the same media-organization skeleton, showing that support reuse is not specific to geography. KGHEdit only needs the KG to expose a compatible shared target.

Shared-slot conflicts. The remaining cases show why heat overlap is not sufficient by itself. In the John Mark Jabalé case in Figure 9, two edits use the same $P39$ title query and the same subject but introduce incompatible targets, Roman Catholic Bishop of Menevia and consul. The shared context is valuable for detecting interaction, but it should not be used as reinforcing evidence. KGHEdit therefore keeps the shared skeleton visible while the balance term prevents incompatible title support from being committed together.

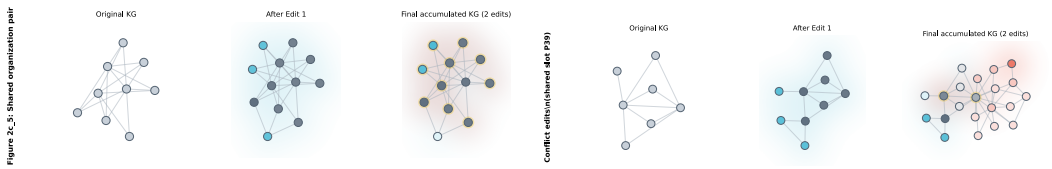


Figure 9: Organization reuse and slot conflict. Left: two ownership edits share Cogeco Media. Right: two John Mark Jabalé title edits share $P39$ but introduce incompatible targets.

Figure 10 replays the same John Mark Jabalé conflict under KGHEdit full and NO-BALANCE. Because the shared skeleton appears in both settings, the conflict is not a visualization artifact or a retrieval side effect. The comparison isolates the role of balance: without it, the competing branch can grow through the same subject-relation slot instead of being filtered before parameter writing.

The Pittsburg $P6$ case in Figure 10 gives the same pattern in an administrative relation. The local KG contains a central administrative skeleton plus competing governance branches. KGHEdit applies the same rule outside the person-title setting: shared-slot overlap is treated as conflict risk, while noncompeting administrative context can still be retained as useful support.

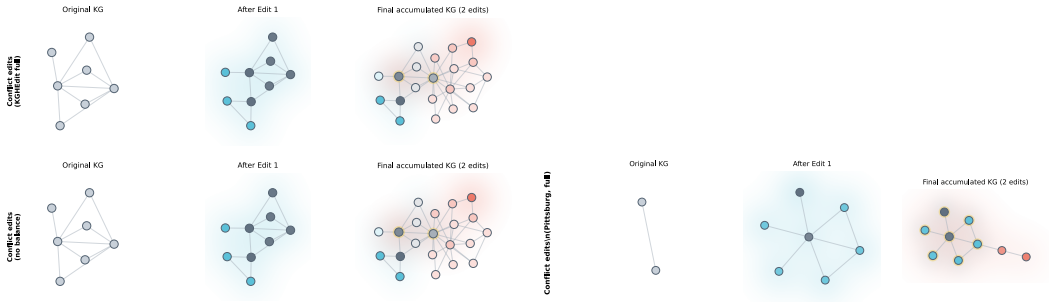


Figure 10: Conflict-balance cases. Left: the same John Mark Jabalé conflict under KGHEdit full and NO-BALANCE. Right: a Pittsburg $P6$ administrative conflict shows the same pattern in another relation family.

Stress-flow robustness. The sensitivity figure in the main text is built from nine settings on a single mixed stress flow. The scan uses $\alpha \in \{0.00, 0.20, 0.80\}$, pair-compatibility thresholds in $\{0.00, 0.30, 0.80\}$, and memory-support gates in $\{0.00, 0.10, 0.40\}$, with the middle value in each set matching the default setting. All settings complete $96/96$ edits. Table 12 reports the final metrics behind Figure 5. Pair compatibility and the memory-support gate return to the default final scores across the scanned range, while $\alpha = 0.80$ gives a very small rewrite-side increase with Locality still close to the default point. This supports the main-text claim that the default setting lies in a broad stable region rather than a narrow tuned optimum.

Table 12: Final metrics behind the unified stress-flow robustness figure. The middle value in each parameter group is the default setting used in the main experiments.

Parameter	Value	Official	Update	Rephrase	Personas	Locality	Prop.
Heat-memory mixing α	0.00	0.6036	0.8289	0.7491	0.6299	0.4742	0.3356
	0.20	**0.6036**	**0.8289**	**0.7491**	**0.6299**	**0.4742**	**0.3356**
	0.80	0.6047	0.8209	0.7574	0.6465	0.4633	0.3356
Pair-compatibility threshold	0.00	0.6036	0.8289	0.7491	0.6299	0.4742	0.3356
	0.30	**0.6036**	**0.8289**	**0.7491**	**0.6299**	**0.4742**	**0.3356**
	0.80	0.6036	0.8289	0.7491	0.6299	0.4742	0.3356
Memory-support gate	0.00	0.6036	0.8289	0.7491	0.6299	0.4742	0.3356
	0.10	**0.6036**	**0.8289**	**0.7491**	**0.6299**	**0.4742**	**0.3356**
	0.40	0.6036	0.8289	0.7491	0.6299	0.4742	0.3356

Efficiency details. Table 13 gives the same matched efficiency rows as Table 4. The reported wall-clock includes graph anchoring, heat solving, support selection, conflict balancing, and the base-editor update in a single sequential loop. This setting deliberately counts the extra work from KGHEdit's larger structured edit batch. A practical deployment can separate lower-cost graph-side planning from the expensive parameter write: incoming edits can be accumulated in a queue, support sets can be prepared in the background without blocking inference, and the resulting edit batches can be applied together during idle periods or on a separate model replica. The short $B20/R20/N400$ stream has the largest relative overhead at $1.61\times$; the two long-horizon protected settings cost $1.35\times$–$1.36\times$ wall-clock time per edit under this conservative sequential measurement while delivering the multi-axis gains reported in the main table.

Table 13: Expanded efficiency details for matched NAS runs.

Setting	Method	Sec/Edit	Relative time
B20/R20/N400	NAS	7.296	$1.00\times$
B20/R20/N400	NAS + KGHEdit	11.737	$1.61\times$
B100/R20/N2000	NAS	6.764	$1.00\times$
B100/R20/N2000	NAS + KGHEdit	9.203	$1.36\times$
B100/R30/N3000	NAS	6.477	$1.00\times$
B100/R30/N3000	NAS + KGHEdit	8.715	$1.35\times$

E PROOFS

The proofs are organized in the same dependency order as the method. We first establish that the graph heat field is a unique, bounded, and smooth diffusion solution. We then connect this field to the local response of a base editor. Finally, we prove the two multi-source results used by KGHEdit to reinforce compatible history and attenuate incompatible overlap before parameter writing.

E.1 STABLE DIFFUSION PROOF

Proof of Proposition 1. We prove existence, uniqueness, and the variational characterization from standard graph Laplacian and graph-diffusion arguments used in manifold regularization, local-global consistency, and diffusion kernels Zhou et al. (2003); Belkin et al. (2006); Kondor & Lafferty (2002); Gasteiger et al. (2019). Let $n = |V_t|$, and let $W_t \in \mathbb{R}^{n\times n}$ be the symmetrized nonnegative affinity matrix used by the diffusion operator. Thus $W_t = W_t^\top$, $(W_t)_{ij} \geq 0$, and $D_t = \mathrm{diag}(d_1, \ldots, d_n)$,

where $d_i = \sum_j (W_t)_{ij}$. For any vector $a \in \mathbb{R}^n$, expand the Laplacian quadratic form:

$$\begin{aligned} a^\top L_t a &= a^\top (D_t - W_t) a \\ &= \sum_i d_i a_i^2 - \sum_{i,j} (W_t)_{ij} a_i a_j \\ &= \frac{1}{2} \sum_{i,j} (W_t)_{ij} a_i^2 + \frac{1}{2} \sum_{i,j} (W_t)_{ij} a_j^2 - \sum_{i,j} (W_t)_{ij} a_i a_j \\ &= \frac{1}{2} \sum_{i,j=1}^{n} (W_t)_{ij} (a_i - a_j)^2 \geq 0. \end{aligned} \tag{10}$$

The third line uses $d_i = \sum_j (W_t)_{ij}$ and symmetry of W_t. Hence L_t is positive semidefinite, even if the local graph is disconnected. The shift by μI removes the null space of L_t. Indeed, for any nonzero a,

$$a^\top (L_t + \mu I) a = a^\top L_t a + \mu \|a\|_2^2 \geq \mu \|a\|_2^2 > 0. \tag{11}$$

Therefore $\Phi_t = L_t + \mu I$ is symmetric positive definite. By the spectral theorem, if $L_t = V \Lambda V^\top$, with orthonormal $V = [v_1, \ldots, v_n]$ and eigenvalues $\lambda_k \geq 0$, then

$$\Phi_t = V(\Lambda + \mu I) V^\top, \qquad \lambda_{\min}(\Phi_t) \geq \mu > 0. \tag{12}$$

Thus Φ_t is invertible, and for every source $q \in \mathbb{R}^n$ the linear system $\Phi_t u = q$ has the unique solution $u = \Phi_t^{-1} q$.

It remains to show that this solution is exactly the minimizer of the stated energy. Let $\mathcal{U}_t = \{\{i, j\} : (W_t)_{ij} > 0,\ i < j\}$ be the undirected support induced by the symmetrized affinity matrix. The edge sum in Proposition 1 is over this undirected support; equivalently, one may write it as one half of the ordered-pair sum above. Define

$$F(u) = \frac{1}{2} \sum_{\{i,j\} \in \mathcal{U}_t} (W_t)_{ij} (u_i - u_j)^2 + \frac{\mu}{2} \|u\|_2^2 - q^\top u, \tag{13}$$

which is the usual graph smoothness penalty plus Tikhonov damping and a linear source term Zhou et al. (2003); Belkin et al. (2006). For coordinate r, the derivative of the smoothness term is

$$\begin{aligned} \frac{\partial}{\partial u_r} \frac{1}{2} \sum_{\{i,j\} \in \mathcal{U}_t} (W_t)_{ij} (u_i - u_j)^2 &= \sum_j (W_t)_{rj} (u_r - u_j) \\ &= d_r u_r - \sum_j (W_t)_{rj} u_j \\ &= (L_t u)_r. \end{aligned} \tag{14}$$

Thus the full gradient and Hessian are

$$\nabla F(u) = (L_t + \mu I) u - q. \tag{15}$$

$$\nabla^2 F(u) = L_t + \mu I = \Phi_t. \tag{16}$$

Since Φ_t is positive definite, F is strictly convex and coercive: $F(u) \to \infty$ as $\|u\|_2 \to \infty$. Hence it has a unique global minimizer. The first-order optimality condition $\nabla F(u^\star) = 0$ is exactly $\Phi_t u^\star = q$, so the minimizer is $u^\star = \Phi_t^{-1} q$.

The same spectral decomposition gives an explicit stability bound:

$$\begin{aligned} u^\star &= \sum_{k=1}^{n} \frac{v_k^\top q}{\lambda_k + \mu} v_k, \\ \|u^\star\|_2^2 &= \sum_{k=1}^{n} \frac{(v_k^\top q)^2}{(\lambda_k + \mu)^2} \leq \frac{1}{\mu^2} \sum_{k=1}^{n} (v_k^\top q)^2 = \frac{\|q\|_2^2}{\mu^2}. \end{aligned} \tag{17}$$

Therefore $\|u^\star\|_2 \leq \|q\|_2 / \mu$. This bound makes the role of μ explicit: damping keeps the heat field finite, removes the constant-component ambiguity of graph Laplacians, and limits the magnitude of the edit scope. Spectrally, each graph-frequency component $v_k^\top q$ is multiplied by $1/(\lambda_k + \mu)$, so high-Laplacian-frequency components are suppressed more strongly. This is the same smoothing principle behind graph-based semi-supervised learning and diffusion kernels Zhou et al. (2003); Kondor & Lafferty (2002); Gasteiger et al. (2019), and it justifies using the solution as a bounded, smooth influence field rather than as an unconstrained spread of the edit. □

E.2 HEAT-GUIDED RESPONSE PROOF

Detailed proof of Theorem 1. Fix an edit step t and write $\theta = \theta_{t-1}$. Suppose the selected support candidates are indexed by $i = 1, \dots, s$. For support weights $\mathbf{w} \in \mathbb{R}^s$, define the weighted support gradient

$$G(\mathbf{w}) = \sum_{i=1}^{s} w_i g_i, \qquad g_i = \nabla_\theta \ell_i(\theta). \tag{18}$$

The vector $G(\mathbf{w})$ is the direction that the support batch presents to the base editor. The local first-order editor assumption used in the theorem means that, in the small-update neighborhood considered by locate-and-edit and batched editing methods, the parameter change can be decomposed as

$$\Delta\theta(\mathbf{w}) = -\eta P_{\theta,t} G(\mathbf{w}) + r(\mathbf{w}), \qquad \|r(\mathbf{w})\|_2 = O(\|\Delta\theta(\mathbf{w})\|_2^2). \tag{19}$$

Here $P_{\theta,t}$ is the editor's first-order projection, preconditioner, or scaling operator, and $r(\mathbf{w})$ is the higher-order residual from optimizer dynamics, value optimization, or changes in the editor projection during the update. This is an analysis model only: KGHEdit neither computes nor estimates $P_{\theta,t}$.

For each tracked prompt z, let $\delta_z(\theta)$ be the logit preference of the tracked answer over its alternative under parameters θ. Define the post-edit preference change

$$\Delta_z(\mathbf{w}) = \delta_z(\theta + \Delta\theta(\mathbf{w})) - \delta_z(\theta). \tag{20}$$

Because δ_z is twice differentiable in the local neighborhood, Taylor's theorem with remainder gives

$$\Delta_z(\mathbf{w}) = \nabla_\theta \delta_z(\theta)^\top \Delta\theta(\mathbf{w}) + \frac{1}{2} \Delta\theta^\top \nabla_\theta^2 \delta_z(\theta + \xi_z \Delta\theta) \Delta\theta \tag{21}$$

for some $\xi_z \in (0,1)$, where $\Delta\theta$ abbreviates $\Delta\theta(\mathbf{w})$ inside the quadratic remainder. Since the tracked prompt set is finite and the Hessians are continuous, their operator norms are bounded on this neighborhood. Hence the Taylor remainder is $O(\|\Delta\theta(\mathbf{w})\|_2^2)$. Substituting Eq. 19 into the linear term gives

$$\begin{aligned} \Delta_z(\mathbf{w}) &= -\eta \nabla_\theta \delta_z(\theta)^\top P_{\theta,t} G(\mathbf{w}) + \nabla_\theta \delta_z(\theta)^\top r(\mathbf{w}) + O(\|\Delta\theta\|_2^2) \\ &= \eta \sum_i \left[-\nabla_\theta \delta_z(\theta)^\top P_{\theta,t} g_i \right] w_i + O(\|\Delta\theta\|_2^2) \\ &= \eta \widehat{\Delta}_z(\mathbf{w}) + O(\|\Delta\theta\|_2^2). \end{aligned} \tag{22}$$

The second line expands $G(\mathbf{w}) = \sum_i w_i g_i$. The third line uses the definition

$$\widehat{\Delta}_z(\mathbf{w}) = \sum_i \psi_i(z) w_i, \qquad \psi_i(z) = -\nabla_\theta \delta_z(\theta)^\top P_{\theta,t} g_i.$$

The residual term $\nabla \delta_z(\theta)^\top r(\mathbf{w})$ is second order because $r(\mathbf{w}) = O(\|\Delta\theta(\mathbf{w})\|_2^2)$ and $\nabla \delta_z(\theta)$ is bounded on the same local neighborhood.

Now partition the tracked prompts into four finite groups: $\mathcal{Z}_t^{\mathrm{tar}}$ for the target prompt, $\mathcal{Z}_t^{\mathrm{com}}$ for compatible prompts, $\mathcal{Z}_t^{\mathrm{loc}}$ for locality prompts, and $\mathcal{Z}_t^{\mathrm{conf}}$ for competing prompts. For each $g \in \{\mathrm{tar}, \mathrm{com}, \mathrm{loc}, \mathrm{conf}\}$, define the average post-edit change and average first-order change by

$$\bar{\Delta}_g(\mathbf{w}) = \frac{1}{|\mathcal{Z}_t^g|} \sum_{z \in \mathcal{Z}_t^g} \Delta_z(\mathbf{w}), \qquad \bar{\widehat{\Delta}}_g(\mathbf{w}) = \frac{1}{|\mathcal{Z}_t^g|} \sum_{z \in \mathcal{Z}_t^g} \widehat{\Delta}_z(\mathbf{w}). \tag{23}$$

Because each group is finite, averaging the promptwise expansion preserves the same remainder order, so for every group g,

$$\bar{\Delta}_g(\mathbf{w}) = \eta \, \bar{\widehat{\Delta}}_g(\mathbf{w}) + O(\|\Delta\theta(\mathbf{w})\|_2^2). \tag{24}$$

The theorem defines the heat-balance separation as

$$s_t(\mathbf{w}) = \min\left\{ \bar{\widehat{\Delta}}_{\mathrm{tar}}(\mathbf{w}), \bar{\widehat{\Delta}}_{\mathrm{com}}(\mathbf{w}), -\bar{\widehat{\Delta}}_{\mathrm{conf}}(\mathbf{w}) \right\} - \left| \bar{\widehat{\Delta}}_{\mathrm{loc}}(\mathbf{w}) \right|.$$

Define the post-edit separation by

$$s_t^{\mathrm{post}}(\mathbf{w}) = \min\left\{ \bar{\Delta}_{\mathrm{tar}}(\mathbf{w}), \bar{\Delta}_{\mathrm{com}}(\mathbf{w}), -\bar{\Delta}_{\mathrm{conf}}(\mathbf{w}) \right\} - \left| \bar{\Delta}_{\mathrm{loc}}(\mathbf{w}) \right|.$$

Eq. 24 implies, for each group,

$$\bar{\Delta}_{\mathrm{tar}}(\mathbf{w}) = \eta\,\bar{\tilde{\Delta}}_{\mathrm{tar}}(\mathbf{w}) + O(\|\Delta\theta(\mathbf{w})\|_2^2), \quad \bar{\Delta}_{\mathrm{com}}(\mathbf{w}) = \eta\,\bar{\tilde{\Delta}}_{\mathrm{com}}(\mathbf{w}) + O(\|\Delta\theta(\mathbf{w})\|_2^2),$$

$$-\bar{\Delta}_{\mathrm{conf}}(\mathbf{w}) = \eta\Big(-\bar{\tilde{\Delta}}_{\mathrm{conf}}(\mathbf{w})\Big) + O(\|\Delta\theta(\mathbf{w})\|_2^2), \quad \big|\bar{\Delta}_{\mathrm{loc}}(\mathbf{w})\big| \le \eta\,\Big|\bar{\tilde{\Delta}}_{\mathrm{loc}}(\mathbf{w})\Big| + O(\|\Delta\theta(\mathbf{w})\|_2^2).$$

Taking the minimum of the first three relations and subtracting the locality term gives

$$s_t^{\mathrm{post}}(\mathbf{w}) \ge \eta\, s_t(\mathbf{w}) - O(\|\Delta\theta(\mathbf{w})\|_2^2),$$

which is Eq. 5 and proves the theorem.

The proof also shows why Eq. 3 is aligned with the theorem. Heat scores choose where the edit should act, Compat increases support that should move together with the target under Proposition 2, and Risk downweights support that would raise competing branches under Proposition 3. Therefore the selected batch is designed to enlarge the first-order separation $s_t(\mathbf{w})$ before the base editor writes parameters. □

E.3 Reinforcement proof

Proof of Proposition 2. Fix edit step t, let $n = |V_t|$, and write $\Phi_t = L_t + \mu I$. The mixed source defined in the main text can be decomposed into the current source and projected previous-edit sources, giving the multi-source heat input

$$q_t^{\mathrm{multi}} = (1-\alpha)q_t^{\mathrm{cur}} + \alpha \sum_{j<t} \omega_{j,t} h_{j\to t}. \tag{25}$$

Thus q_t^{multi} represents several heat-source terms placed on the same local graph, rather than a single replay example. Consider the corresponding heat energy

$$\begin{aligned} F_t^{\mathrm{multi}}(u) &= \frac{1}{2}\sum_{\{i,j\}\in\mathcal{U}_t} (W_t)_{ij}(u_i-u_j)^2 + \frac{\mu}{2}\|u\|_2^2 - (q_t^{\mathrm{multi}})^\top u \\ &= \frac{1}{2}\sum_{\{i,j\}\in\mathcal{U}_t} (W_t)_{ij}(u_i-u_j)^2 + \frac{\mu}{2}\|u\|_2^2 - (1-\alpha)(q_t^{\mathrm{cur}})^\top u - \alpha\sum_{j<t}\omega_{j,t}(h_{j\to t})^\top u. \end{aligned} \tag{26}$$

This smoothness-plus-source form is the standard variational view used in graph regularization and diffusion kernels Zhou et al. (2003); Belkin et al. (2006); Kondor & Lafferty (2002). Here $\mathcal{U}_t = \{\{i,j\} : (W_t)_{ij} > 0,\ i<j\}$ is the undirected support of the symmetrized local affinity matrix, as in the proof of Proposition 1. The first two terms are shared by all sources: they enforce one smooth and damped field over $\mathcal{G}_t$. The last line shows that the current edit and all retained historical edits enter only as heat-source terms. Thus the mathematical object is a multi-source diffusion energy on a common KG, not a collection of independent point edits.

We now derive the equilibrium. For any $a \in \mathbb{R}^n$,

$$a^\top \Phi_t a = a^\top L_t a + \mu\|a\|_2^2 \ge \mu\|a\|_2^2 > 0 \qquad (a \ne 0), \tag{27}$$

because L_t is positive semidefinite and $\mu > 0$. Hence Φ_t is symmetric positive definite. Therefore F_t^{multi} is strictly convex and coercive, so it has a unique global minimizer. Taking the coordinate derivative of the smoothness term exactly as in Proposition 1 gives

$$\frac{\partial}{\partial u_r}\frac{1}{2}\sum_{\{i,j\}\in\mathcal{U}_t}(W_t)_{ij}(u_i-u_j)^2 = (L_t u)_r. \tag{28}$$

The derivative of the damping term is μu_r, and the derivative of the multi-source linear term is $-(q_t^{\mathrm{multi}})_r$. Thus

$$\nabla F_t^{\mathrm{multi}}(u) = (L_t+\mu I)u - q_t^{\mathrm{multi}} = \Phi_t u - q_t^{\mathrm{multi}}. \tag{29}$$

The first-order optimality condition for the unique minimizer $u_t^\star$ is therefore

$$\Phi_t u_t^\star = q_t^{\mathrm{multi}}, \tag{30}$$

which is the multi-source heat equilibrium stated in Proposition 2.

It remains to derive the balance identity. Let u_t^{cur} and $u_{j\to t}$ be the single-source equilibria

$$\begin{aligned}\Phi_t u_t^{\mathrm{cur}} &= q_t^{\mathrm{cur}},\\ \Phi_t u_{j\to t} &= h_{j\to t}.\end{aligned} \tag{31}$$

Define

$$\tilde{u}_t = (1-\alpha)u_t^{\mathrm{cur}} + \alpha \sum_{j<t} \omega_{j,t} u_{j\to t}. \tag{32}$$

Applying the same diffusion operator Φ_t to this candidate field gives

$$\begin{aligned}\Phi_t \tilde{u}_t &= (1-\alpha)\Phi_t u_t^{\mathrm{cur}} + \alpha \sum_{j<t} \omega_{j,t} \Phi_t u_{j\to t}\\ &= (1-\alpha) q_t^{\mathrm{cur}} + \alpha \sum_{j<t} \omega_{j,t} h_{j\to t} = q_t^{\mathrm{multi}}.\end{aligned} \tag{33}$$

Since the multi-source equilibrium $\Phi_t u = q_t^{\mathrm{multi}}$ is unique, $\tilde{u}_t = u_t^\star$. This proves Eq. 32. In other words, the additive expression is not an extra modeling assumption; it follows from the uniqueness of the common heat equilibrium under multiple sources.

We next justify why this heat equilibrium supports the multi-edit interpretation in the main text. Let $R_t = D_t + \mu I$, so R_t is diagonal with strictly positive entries $d_i + \mu$, and

$$\Phi_t = L_t + \mu I = D_t - W_t + \mu I = R_t - W_t = R_t(I - R_t^{-1} W_t). \tag{34}$$

Define $\Pi_t = R_t^{-1} W_t$. The i-th row sum of Π_t is

$$\sum_j (\Pi_t)_{ij} = \frac{1}{d_i + \mu} \sum_j (W_t)_{ij} = \frac{d_i}{d_i + \mu} < 1. \tag{35}$$

Thus $\|\Pi_t\|_\infty < 1$, so the spectral radius is also below one. The Neumann series is therefore valid:

$$\Phi_t^{-1} = (I - \Pi_t)^{-1} R_t^{-1} = \sum_{k=0}^{\infty} \Pi_t^k R_t^{-1}. \tag{36}$$

Every factor in this series is entrywise nonnegative because $W_t \geq 0$, $R_t^{-1} \geq 0$, and powers of a nonnegative matrix remain nonnegative. Hence $\Phi_t^{-1} \geq 0$ entrywise. Consequently, nonnegative edit sources produce nonnegative heat fields. The series also gives a path-based interpretation: the k-th term aggregates length-k walks in the local graph, with a damping factor from R_t^{-1} at each step. This is precisely the local smoothing behavior used in graph diffusion methods Kondor & Lafferty (2002); Gasteiger et al. (2019): influence can move through graph paths, but the row-sum bound keeps the accumulated mass finite.

Now consider the three cases claimed in the method section. First, if a historical edit has no projected support on the current local graph, then its projected source is zero, $h_{j\to t} = 0$, by the normalization convention used for projected memory. Its term in Eq. 32 is therefore exactly zero:

$$\omega_{j,t} u_{j\to t} = \omega_{j,t} \Phi_t^{-1} h_{j\to t} = 0. \tag{37}$$

Second, if the edit is old, the accumulated decay weight $\omega_{j,t}$ multiplies its whole contribution. By Proposition 1,

$$\|\omega_{j,t} u_{j\to t}\|_2 \leq \frac{\omega_{j,t}}{\mu} \|h_{j\to t}\|_2, \tag{38}$$

so damping and memory decay jointly bound the magnitude of every historical source. Summing the same bound over all sources gives

$$\|u_t^\star\|_2 \leq \frac{1}{\mu} \left((1-\alpha)\|q_t^{\mathrm{cur}}\|_2 + \alpha \sum_{j<t} \omega_{j,t} \|h_{j\to t}\|_2 \right), \tag{39}$$

which shows that the balanced field remains finite as long as the retained memory mass is finite.

Third, overlapping compatible edits reinforce rather than cancel. Since $\Phi_t^{-1} \geq 0$, if two nonnegative sources place mass on nearby or shared nodes, their heat fields are nonnegative and are added with

nonnegative coefficients in Eq. 32. For any retained memory candidate e with node set $Z(e)$, its heat score is linear in $u_t^\star$:

$$\begin{aligned}\mathrm{Heat}_e(u_t^\star) &= \frac{1}{|Z(e)\cap V_t|}\sum_{v\in Z(e)\cap V_t} u_t^\star(v)\\ &= (1-\alpha)\mathrm{Heat}_e(u_t^{\mathrm{cur}}) + \alpha\sum_{j<t}\omega_{j,t}\mathrm{Heat}_e(u_{j\to t}),\end{aligned} \tag{40}$$

whenever $Z(e)\cap V_t \neq \emptyset$, and it is zero otherwise. Thus the diffusion field does not itself decide semantic compatibility; it exposes where current and historical edits overlap. The $\mathrm{Compat}(c, r_t)$ and $\mathrm{Risk}(c, r_t)$ terms in Eq. 3 then decide whether this overlap should reinforce the update or be treated as a conflict before any model weights are changed. This proves the compatible multi-source reinforcement claim. □

E.4 Conflict suppression proof

Proof of Proposition 3. We now prove the graph-side conflict attenuation property used by KGHEdit: when two KG-anchored support branches are incompatible, a positive coupling penalty reduces their simultaneous heat activation before examples are sent to the base editor. The joint heat $J_\gamma(c_a, c_b)$ in Proposition 3 is formalized below as masked overlap on the local KG.

Let $\Phi = \Phi_t = L_t + \mu I$. By Proposition 1, Φ is symmetric positive definite. Let p_{c_a} and p_{c_b} be the normalized branch sources for two incompatible candidates c_a, c_b, and let $M = M_{ab} \succeq 0$ be the conflict mask. The formal coupled objective is

$$\min_{v_a,v_b}\frac{1}{2}v_a^\top\Phi v_a - p_{c_a}^\top v_a + \frac{1}{2}v_b^\top\Phi v_b - p_{c_b}^\top v_b + \gamma v_a^\top M v_b, \qquad \gamma\geq 0. \tag{41}$$

Stacking the two fields as $z = [v_a^\top, v_b^\top]^\top$, the Hessian of Eq. 41 is

$$\mathcal{H}_\gamma = \begin{bmatrix}\Phi & \gamma M\\ \gamma M & \Phi\end{bmatrix}. \tag{42}$$

At $\gamma = 0$, $\mathcal{H}_0 = \mathrm{diag}(\Phi, \Phi) \succ 0$. Eigenvalues depend continuously on γ; hence there exists $\epsilon > 0$ such that $\mathcal{H}_\gamma \succ 0$ for all $0 \leq \gamma < \epsilon$. A sufficient explicit condition is $\gamma\|M\|_2 < \lambda_{\min}(\Phi)$, because the off-diagonal block perturbation has spectral norm $\gamma\|M\|_2$. Under this condition, Eq. 41 is strictly convex and has a unique minimizer $(v_a(\gamma), v_b(\gamma))$.

The first-order conditions are

$$\begin{aligned}\Phi v_a(\gamma) + \gamma M v_b(\gamma) &= p_{c_a},\\ \Phi v_b(\gamma) + \gamma M v_a(\gamma) &= p_{c_b}.\end{aligned} \tag{43}$$

At $\gamma = 0$, the solution is the pair of independent heat fields

$$v_a^0 = \Phi^{-1}p_{c_a}, \qquad v_b^0 = \Phi^{-1}p_{c_b}. \tag{44}$$

Because $\mathcal{H}_\gamma$ is nonsingular near zero, the implicit function theorem gives differentiable solution paths in γ. Differentiating Eq. 43 and evaluating at $\gamma = 0$ yields

$$\begin{aligned}\Phi\dot v_a(0) + M v_b^0 &= 0,\\ \Phi\dot v_b(0) + M v_a^0 &= 0,\end{aligned} \quad\Rightarrow\quad \begin{aligned}\dot v_a(0) &= -\Phi^{-1}M v_b^0,\\ \dot v_b(0) &= -\Phi^{-1}M v_a^0.\end{aligned} \tag{45}$$

Now define the incompatible overlap, which is the formal joint heat $J_\gamma(c_a, c_b)$ used in the main text,

$$J_\gamma(c_a, c_b) = \Omega_{ab}(\gamma) = v_a(\gamma)^\top M v_b(\gamma). \tag{46}$$

Its derivative at zero is

$$\begin{aligned}\Omega'_{ab}(0) &= \dot v_a(0)^\top M v_b^0 + (v_a^0)^\top M\dot v_b(0)\\ &= -(Mv_b^0)^\top\Phi^{-1}(Mv_b^0) - (Mv_a^0)^\top\Phi^{-1}(Mv_a^0) \leq 0.\end{aligned} \tag{47}$$

The inequality follows because $\Phi^{-1} \succ 0$. It is strict whenever at least one source has nonzero mass on the conflict mask. Therefore, for small positive γ,

$$\Omega_{ab}(\gamma) = \Omega_{ab}(0) + \gamma\Omega'_{ab}(0) + O(\gamma^2) \leq \Omega_{ab}(0) + O(\gamma^2), \tag{48}$$

and the first-order effect of the conflict term is to reduce the simultaneous activation of the incompatible branches. This proves the attenuation statement.

Finally, we connect this ideal coupled objective to the lightweight implementation in Eq. 3. Solving Eq. 41 for every possible candidate pair would be expensive. KGHEdit instead uses ordinary heat diffusion to expose overlap and uses $\mathrm{Risk}(\cdot, r_t)$ as a computationally cheap proxy for the masked conflict term. For a current candidate c, if

$$\lambda_r \mathrm{Risk}(c, r_t) > \mathrm{Heat}(c) + \lambda_c \mathrm{Compat}(c, r_t), \tag{49}$$

then $s(c) < 0$, so any positive-score filter removes it, and even under pure top-k ranking it is placed below candidates with the same heat but lower conflict risk. Hence the implementation follows the theoretical principle: heat identifies where edits interact, compatibility keeps reinforcing support, and risk suppresses incompatible branches before parameter editing. □